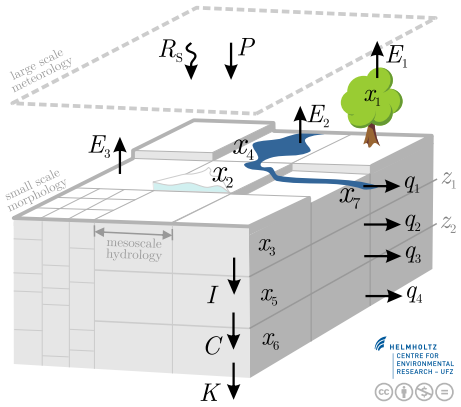
The mesoscale Hydrologic Model

mHM



Documentation for version 5.10

Edited by Luis Samaniego

in cooperation with

Maren Kaluza, Rohini Kumar, Oldrich Rakovec,
Lennart Schüler, Robert Schweppe, Pallav Shrestha, Stephan Thober

HELMHOLTZ CENTRE FOR ENVIRONMENTAL RESEARCH – UFZ

Former mHM project collaborators

Johannes Brenner, John Craven, Cüneyd M. Demirel, Matthias Cuntz, Giovanni Dalmasso, Miao Jing, Ben Langenberg, Juliane Mai, Jude Musuuza, Vladyslav Prykhodko, David Schäfer, Christoph Schneider, Diana Spieler, Simon Stisen, Martin Schrön, Matthias Zink

The mHM project

www.ufz.de/mhm

Contents

1 Introduction to mHM **1**

1.1 Short Description . . . 1
1.2 The Grid-based mHM Model . . . 1
1.3 Model Formulation . . . 3
1.4 The Multiscale Parameter Regionalization Technique . . . 4
1.5 The Parameter Estimation Problem . . . 5
1.6 Model Calibration . . . 6
1.7 Helpful links . . . 7
1.8 Test basin . . . 7
1.9 Protocols . . . 7

2 Getting Started **9**

2.1 Receive mHM . . . 9
2.2 Install NETCDF . . . 9
2.2.1 Short Guide Install to NETCDF on Linux (example Ubuntu 16.04 with gfortran v5.4.0) . . . 9
2.2.2 Short Guide Install to NETCDF on MacOS . . . 10
2.3 Run mHM under CYGWIN on Windows . . . 11
2.4 Compile mHM . . . 14
2.5 Test mHM on Example Basin . . . 15
2.6 Run your own Simulation . . . 15
2.6.1 Main Configuration: Paths, Periods, Switches . . . 15
2.6.2 Output Configuration: Time Steps, States, Fluxes . . . 16
2.6.3 Regionalised Parameters: Initial Values and Ranges . . . 16
2.7 Calibration and Optimization . . . 16
2.7.1 The Optimization Routines . . . 16
2.7.2 Calibration Settings for mHM . . . 17
2.7.3 Final Calibration Results . . . 17

3 Data Preparation for mHM **19**

3.1 Getting Started . . . 19
3.1.1 Meteorological variables . . . 19
3.1.2 Morphological variables . . . 20

3.1.3 Land Cover . . . 20
3.1.4 Gauging Station Information . . . 20
3.1.5 Optional Data . . . 21
3.2 Preparation of the Forcings . . . 21
3.2.1 Extract Data to the Size of a Catchment . . . 21
3.2.2 Extact Data to the Time Period of Interest . . . 21
3.2.3 Data Types . . . 21
3.2.4 Variable Names . . . 22
3.2.5 The Header File . . . 22
3.2.6 NODATA values . . . 22
3.3 Preparation of the LatLon Grid . . . 22
3.4 Preparation of the Morphological Data . . . 23
3.5 A possible GIS workflow . . . 24
3.5.1 General considerations . . . 24
3.5.2 Slope map . . . 27
3.5.3 Aspect map . . . 27
3.5.4 Fill DEM sinks . . . 28
3.5.5 Flow direction and flow accumulation . . . 28
3.5.5.1 Spatial Analyst . . . 28
3.5.5.2 Arc Hydro Tools . . . 29
3.5.6 Gauges map . . . 30
3.5.7 Watershed delineation . . . 32
3.5.8 Mask the datasets . . . 33
3.5.9 Write the ascii grids . . . 33
3.6 Land Cover Data . . . 33
3.7 Table Data . . . 34
3.7.1 The soil look-up table . . . 34
3.7.2 The hydrogeolgy look-up table . . . 34
3.7.3 The LAI look-up table . . . 35
3.7.4 The gauge files . . . 35
3.8 Post-GIS preparation . . . 36

4 Visualizing Model Output **37**

5 Calibration Options **39**

5.1 Calibration of Parameters on Observations . . . 39
5.1.1 Parameters . . . 39
5.1.2 Methods . . . 39
5.1.3 Functions . . . 39
5.1.4 Data . . . 39
5.1.5 Output . . . 40

6 Coding and Documentation Style 41

7 The details about the test basin 45

8 Protocols for setting up a new mHM basin 47

8.1 Check for possible input data errors using mHM outputs . . . 47

8.2 Protocol for generating a restart file . . . 48

8.3 Protocol for determining the warm-up period for a new basin . . . 48

9 mRM - the Multiscale Routing Model: Running the stand-alone version of the routing model 49

9.1 Introduction to mRM . . . 49

9.1.1 Finite Difference Approximation of Kinematic Wave Equation . . . 49

9.1.2 Stream Celerity Parametrization based on Terrain Slope . . . 50

9.1.3 Adaptive Time Step (aTS) Implementation . . . 50

9.1.4 Regionalized Muskingum-Cunge parametrization . . . 51

9.2 Getting Started . . . 51

9.2.1 Compiling mRM . . . 52

9.2.2 Test mRM on Example Basin . . . 52

9.2.3 Run your own Simulation . . . 52

9.2.3.1 Main configuration: Path, Periods, Switches . . . 52

9.2.4 Output Configuration: Time Steps, States, Fluxes . . . 53

9.2.5 Calibration and Optimization . . . 53

9.3 Data preparation for mRM . . . 53

10 mHM Dependencies 55

11 Publications using mHM 57

12 mHM RELEASE NOTES 61

13 Modules Index 71

13.1 Modules List . . . 71

14 Data Type Index 75

14.1 Data Types List . . . 75

15 File Index 79

15.1 File List . . . 79

16 Module Documentation 83

16.1 dummy_mpr Module Reference . . . 83

16.2 dummy_mrm Module Reference . . . 83

16.3 mo_anneal Module Reference . . . 83

16.3.1 Function/Subroutine Documentation . . . 83

16.3.1.1 anneal_dp() . . . 83
16.3.1.2 dchange_dp() . . . 84
16.3.1.3 generate_neighborhood_weight_dp() . . . 85
16.3.1.4 gettemperature_dp() . . . 85
16.3.1.5 pargen_anneal_dp() . . . 86
16.3.1.6 pargen_dds_dp() . . . 87
16.4 mo_append Module Reference . . . 87
16.4.1 Detailed Description . . . 88
16.4.2 Function/Subroutine Documentation . . . 88
16.4.2.1 append_char_3d() . . . 88
16.4.2.2 append_char_m_m() . . . 89
16.4.2.3 append_char_v_s() . . . 89
16.4.2.4 append_char_v_v() . . . 89
16.4.2.5 append_dp_3d() . . . 89
16.4.2.6 append_dp_m_m() . . . 89
16.4.2.7 append_dp_v_s() . . . 89
16.4.2.8 append_dp_v_v() . . . 89
16.4.2.9 append_i4_3d() . . . 90
16.4.2.10 append_i4_m_m() . . . 90
16.4.2.11 append_i4_v_s() . . . 90
16.4.2.12 append_i4_v_v() . . . 90
16.4.2.13 append_i8_3d() . . . 90
16.4.2.14 append_i8_m_m() . . . 90
16.4.2.15 append_i8_v_s() . . . 91
16.4.2.16 append_i8_v_v() . . . 91
16.4.2.17 append_lgt_3d() . . . 91
16.4.2.18 append_lgt_m_m() . . . 91
16.4.2.19 append_lgt_v_s() . . . 91
16.4.2.20 append_lgt_v_v() . . . 91
16.4.2.21 append_sp_3d() . . . 91
16.4.2.22 append_sp_m_m() . . . 92
16.4.2.23 append_sp_v_s() . . . 92
16.4.2.24 append_sp_v_v() . . . 92
16.4.2.25 paste_char_m_m() . . . 92
16.4.2.26 paste_char_m_s() . . . 92
16.4.2.27 paste_char_m_v() . . . 92
16.4.2.28 paste_dp_m_m() . . . 92
16.4.2.29 paste_dp_m_s() . . . 93
16.4.2.30 paste_dp_m_v() . . . 93
16.4.2.31 paste_i4_m_m() . . . 93

16.4.2.32 paste_i4_m_s() . . . 93
16.4.2.33 paste_i4_m_v() . . . 93
16.4.2.34 paste_i8_m_m() . . . 93
16.4.2.35 paste_i8_m_s() . . . 94
16.4.2.36 paste_i8_m_v() . . . 94
16.4.2.37 paste_lgt_m_m() . . . 94
16.4.2.38 paste_lgt_m_s() . . . 94
16.4.2.39 paste_lgt_m_v() . . . 94
16.4.2.40 paste_sp_m_m() . . . 94
16.4.2.41 paste_sp_m_s() . . . 94
16.4.2.42 paste_sp_m_v() . . . 95
16.5 mo_canopy_interc Module Reference . . . 95
16.5.1 Detailed Description . . . 95
16.5.2 Function/Subroutine Documentation . . . 95
16.5.2.1 canopy_interc() . . . 95
16.6 mo_common_constants Module Reference . . . 97
16.6.1 Detailed Description . . . 98
16.6.2 Variable Documentation . . . 98
16.6.2.1 dayhours . . . 98
16.6.2.2 daysecs . . . 98
16.6.2.3 eps_dp . . . 98
16.6.2.4 eps_sp . . . 98
16.6.2.5 hoursecs . . . 98
16.6.2.6 maxnlcovers . . . 99
16.6.2.7 maxnobasins . . . 99
16.6.2.8 ncolpars . . . 99
16.6.2.9 nodata_dp . . . 99
16.6.2.10 nodata_i4 . . . 100
16.6.2.11 p1_initstatefluxes . . . 100
16.6.2.12 yeardays . . . 100
16.6.2.13 yearmonths . . . 100
16.6.2.14 yearmonths_i4 . . . 100
16.7 mo_common_file Module Reference . . . 100
16.7.1 Detailed Description . . . 101
16.7.2 Variable Documentation . . . 101
16.7.2.1 file_config . . . 101
16.7.2.2 file_dem . . . 101
16.7.2.3 uconfig . . . 101
16.7.2.4 udem . . . 102
16.7.2.5 ulcoverclass . . . 102

16.8 mo_common_functions Module Reference . . . 102
16.8.1 Detailed Description . . . 102
16.8.2 Function/Subroutine Documentation . . . 102
16.8.2.1 in_bound() . . . 102
16.9 mo_common_mhm_mrm_file Module Reference . . . 103
16.9.1 Detailed Description . . . 103
16.9.2 Variable Documentation . . . 103
16.9.2.1 file_opti . . . 104
16.9.2.2 file_opti_nml . . . 104
16.9.2.3 uopti . . . 104
16.9.2.4 uopti_nml . . . 104
16.10 mo_common_mhm_mrm_read_config Module Reference . . . 104
16.10.1 Detailed Description . . . 104
16.10.2 Function/Subroutine Documentation . . . 105
16.10.2.1 check_optimization_settings() . . . 105
16.10.2.2 common_check_resolution() . . . 106
16.10.2.3 common_mhm_mrm_read_config() . . . 107
16.11 mo_common_mhm_mrm_variables Module Reference . . . 108
16.11.1 Detailed Description . . . 109
16.11.2 Variable Documentation . . . 109
16.11.2.1 c2tstu . . . 109
16.11.2.2 dds_r . . . 110
16.11.2.3 dirrestartin . . . 110
16.11.2.4 evalper . . . 110
16.11.2.5 lcyearid . . . 110
16.11.2.6 mcmc_error_params . . . 110
16.11.2.7 mcmc_opti . . . 110
16.11.2.8 mrm_coupling_mode . . . 111
16.11.2.9 nerror_model . . . 111
16.11.2.10 niterations . . . 111
16.11.2.11 ntstepday . . . 111
16.11.2.12 opti_function . . . 111
16.11.2.13 opti_method . . . 111
16.11.2.14 optimize . . . 111
16.11.2.15 optimize_restart . . . 112
16.11.2.16 read_restart . . . 112
16.11.2.17 readper . . . 112
16.11.2.18 resolutionrouting . . . 112
16.11.2.19 sa_temp . . . 112
16.11.2.20 sce_ngs . . . 112

16.11.2.21 sce_npg . . . 113
16.11.2.22 sce_nps . . . 113
16.11.2.23 seed . . . 113
16.11.2.24 simper . . . 113
16.11.2.25 timestep . . . 113
16.11.2.26 warmingdays . . . 113
16.11.2.27 warmper . . . 114
16.12 mo_common_read_config Module Reference . . . 114
16.12.1 Detailed Description . . . 114
16.12.2 Function/Subroutine Documentation . . . 114
16.12.2.1 common_read_config() . . . 114
16.12.2.2 set_land_cover_scenes_id() . . . 116
16.13 mo_common_read_data Module Reference . . . 117
16.13.1 Detailed Description . . . 117
16.13.2 Function/Subroutine Documentation . . . 117
16.13.2.1 read_dem() . . . 117
16.13.2.2 read_lcover() . . . 118
16.14 mo_common_restart Module Reference . . . 119
16.14.1 Detailed Description . . . 119
16.14.2 Function/Subroutine Documentation . . . 119
16.14.2.1 read_grid_info() . . . 120
16.14.2.2 write_grid_info() . . . 121
16.15 mo_common_variables Module Reference . . . 122
16.15.1 Detailed Description . . . 123
16.15.2 Variable Documentation . . . 123
16.15.2.1 alma_convention . . . 123
16.15.2.2 contact . . . 123
16.15.2.3 conventions . . . 124
16.15.2.4 dircommonfiles . . . 124
16.15.2.5 dirconfigout . . . 124
16.15.2.6 dirlcover . . . 124
16.15.2.7 dirmorpho . . . 124
16.15.2.8 dirout . . . 124
16.15.2.9 dirrestartout . . . 124
16.15.2.10 filelatlon . . . 125
16.15.2.11 global_parameters . . . 125
16.15.2.12 global_parameters_name . . . 125
16.15.2.13 history . . . 125
16.15.2.14 iflag_cordinate_sys . . . 125
16.15.2.15 l0_basin . . . 125

16.15.2.16 l0_elev . . . 126
16.15.2.17 l0_l11_remap . . . 126
16.15.2.18 l0_l1_remap . . . 126
16.15.2.19 l0_lcover . . . 126
16.15.2.20 l11_basin . . . 126
16.15.2.21 l1_l11_remap . . . 126
16.15.2.22 lc_year_end . . . 126
16.15.2.23 lc_year_start . . . 127
16.15.2.24 lcfilename . . . 127
16.15.2.25 level0 . . . 127
16.15.2.26 level1 . . . 127
16.15.2.27 mhm_details . . . 128
16.15.2.28 nbasins . . . 128
16.15.2.29 nlcoverscene . . . 128
16.15.2.30 nprocesses . . . 128
16.15.2.31 nuniquel0basins . . . 128
16.15.2.32 processmatrix . . . 129
16.15.2.33 project_details . . . 129
16.15.2.34 resolutionhydrology . . . 129
16.15.2.35 setup_description . . . 129
16.15.2.36 simulation_type . . . 129
16.15.2.37 write_restart . . . 129
16.16 mo_constants Module Reference . . . 130
16.16.1 Detailed Description . . . 132
16.16.2 Variable Documentation . . . 132
16.16.2.1 cp0_dp . . . 132
16.16.2.2 cp0_sp . . . 132
16.16.2.3 dayhours . . . 132
16.16.2.4 daysecs . . . 132
16.16.2.5 deg2rad_dp . . . 133
16.16.2.6 deg2rad_sp . . . 133
16.16.2.7 euler . . . 133
16.16.2.8 euler_d . . . 133
16.16.2.9 gravity_dp . . . 133
16.16.2.10 gravity_sp . . . 133
16.16.2.11 nerr . . . 133
16.16.2.12 nin . . . 133
16.16.2.13 nnml . . . 134
16.16.2.14 nout . . . 134
16.16.2.15 p0_dp . . . 134

16.16.2.16 p0_sp . . . 134
16.16.2.17 pi . . . 134
16.16.2.18 pi_d . . . 134
16.16.2.19 pi_dp . . . 134
16.16.2.20 pi_sp . . . 135
16.16.2.21 pio2 . . . 135
16.16.2.22 pio2_d . . . 135
16.16.2.23 pio2_dp . . . 135
16.16.2.24 pio2_sp . . . 135
16.16.2.25 psychro_dp . . . 135
16.16.2.26 psychro_sp . . . 135
16.16.2.27 rad2deg_dp . . . 136
16.16.2.28 rad2deg_sp . . . 136
16.16.2.29 radiusearth_dp . . . 136
16.16.2.30 radiusearth_sp . . . 136
16.16.2.31 rho0_dp . . . 136
16.16.2.32 rho0_sp . . . 136
16.16.2.33 secday_dp . . . 136
16.16.2.34 secday_sp . . . 136
16.16.2.35 sigma_dp . . . 137
16.16.2.36 sigma_sp . . . 137
16.16.2.37 solarconst_dp . . . 137
16.16.2.38 solarconst_sp . . . 137
16.16.2.39 specheatet_dp . . . 137
16.16.2.40 specheatet_sp . . . 137
16.16.2.41 sqrt2 . . . 137
16.16.2.42 sqrt2_d . . . 138
16.16.2.43 sqrt2_dp . . . 138
16.16.2.44 sqrt2_sp . . . 138
16.16.2.45 t0_dp . . . 138
16.16.2.46 t0_sp . . . 138
16.16.2.47 twopi . . . 138
16.16.2.48 twopi_d . . . 138
16.16.2.49 twopi_dp . . . 139
16.16.2.50 twopi_sp . . . 139
16.16.2.51 twothird_dp . . . 139
16.16.2.52 twothird_sp . . . 139
16.16.2.53 yeardays . . . 139
16.16.2.54 yearmonths . . . 139
16.17 mo_corr Module Reference . . . 139

16.17.1 Function/Subroutine Documentation . . . 140
16.17.1.1 arth_dp() . . . 140
16.17.1.2 arth_i4() . . . 141
16.17.1.3 arth_sp() . . . 141
16.17.1.4 autocoeffk_1d_dp() . . . 141
16.17.1.5 autocoeffk_1d_sp() . . . 141
16.17.1.6 autocoeffk_dp() . . . 141
16.17.1.7 autocoeffk_sp() . . . 141
16.17.1.8 autocorr_1d_dp() . . . 142
16.17.1.9 autocorr_1d_sp() . . . 142
16.17.1.10 autocorr_dp() . . . 142
16.17.1.11 autocorr_sp() . . . 142
16.17.1.12 corr_dp() . . . 142
16.17.1.13 corr_sp() . . . 142
16.17.1.14 crosscoeffk_dp() . . . 143
16.17.1.15 crosscoeffk_sp() . . . 143
16.17.1.16 crosscorr_dp() . . . 143
16.17.1.17 crosscorr_sp() . . . 143
16.17.1.18 four1_dp() . . . 143
16.17.1.19 four1_sp() . . . 143
16.17.1.20 fourrow_dp() . . . 144
16.17.1.21 fourrow_sp() . . . 144
16.17.1.22 realft_dp() . . . 144
16.17.1.23 realft_sp() . . . 144
16.17.1.24 swap_1d_dpc() . . . 145
16.17.1.25 swap_1d_spc() . . . 145
16.17.1.26 zroots_unity_dp() . . . 145
16.17.1.27 zroots_unity_sp() . . . 146
16.17.2 Variable Documentation . . . 146
16.17.2.1 npar2_arth . . . 146
16.17.2.2 npar_arth . . . 146
16.18 mo_dds Module Reference . . . 146
16.18.1 Detailed Description . . . 147
16.18.2 Function/Subroutine Documentation . . . 147
16.18.2.1 dds() . . . 147
16.18.2.2 mdds() . . . 149
16.18.2.3 neigh_value() . . . 150
16.19 mo_errormeasures Module Reference . . . 150
16.19.1 Function/Subroutine Documentation . . . 152
16.19.1.1 bias_dp_1d() . . . 152

16.19.1.2 bias_dp_2d() . . . 152
16.19.1.3 bias_dp_3d() . . . 152
16.19.1.4 bias_sp_1d() . . . 153
16.19.1.5 bias_sp_2d() . . . 153
16.19.1.6 bias_sp_3d() . . . 153
16.19.1.7 kge_dp_1d() . . . 153
16.19.1.8 kge_dp_2d() . . . 153
16.19.1.9 kge_dp_3d() . . . 153
16.19.1.10 kge_sp_1d() . . . 153
16.19.1.11 kge_sp_2d() . . . 154
16.19.1.12 kge_sp_3d() . . . 154
16.19.1.13 kgenocorr_dp_1d() . . . 154
16.19.1.14 kgenocorr_dp_2d() . . . 154
16.19.1.15 kgenocorr_dp_3d() . . . 154
16.19.1.16 kgenocorr_sp_1d() . . . 154
16.19.1.17 kgenocorr_sp_2d() . . . 155
16.19.1.18 kgenocorr_sp_3d() . . . 155
16.19.1.19 nnse_dp_1d() . . . 155
16.19.1.20 nnse_dp_2d() . . . 155
16.19.1.21 nnse_dp_3d() . . . 155
16.19.1.22 nnse_sp_1d() . . . 155
16.19.1.23 nnse_sp_2d() . . . 156
16.19.1.24 nnse_sp_3d() . . . 156
16.19.1.25 mae_dp_1d() . . . 156
16.19.1.26 mae_dp_2d() . . . 156
16.19.1.27 mae_dp_3d() . . . 157
16.19.1.28 mae_sp_1d() . . . 157
16.19.1.29 mae_sp_2d() . . . 158
16.19.1.30 mae_sp_3d() . . . 158
16.19.1.31 mse_dp_1d() . . . 158
16.19.1.32 mse_dp_2d() . . . 159
16.19.1.33 mse_dp_3d() . . . 160
16.19.1.34 mse_sp_1d() . . . 160
16.19.1.35 mse_sp_2d() . . . 161
16.19.1.36 mse_sp_3d() . . . 162
16.19.1.37 nse_dp_1d() . . . 162
16.19.1.38 nse_dp_2d() . . . 163
16.19.1.39 nse_dp_3d() . . . 163
16.19.1.40 nse_sp_1d() . . . 163
16.19.1.41 nse_sp_2d() . . . 163

16.19.1.42 nse_sp_3d() . . . 163
16.19.1.43 rmse_dp_1d() . . . 163
16.19.1.44 rmse_dp_2d() . . . 164
16.19.1.45 rmse_dp_3d() . . . 164
16.19.1.46 rmse_sp_1d() . . . 165
16.19.1.47 rmse_sp_2d() . . . 165
16.19.1.48 rmse_sp_3d() . . . 165
16.19.1.49 sae_dp_1d() . . . 166
16.19.1.50 sae_dp_2d() . . . 166
16.19.1.51 sae_dp_3d() . . . 167
16.19.1.52 sae_sp_1d() . . . 168
16.19.1.53 sae_sp_2d() . . . 168
16.19.1.54 sae_sp_3d() . . . 169
16.19.1.55 sse_dp_1d() . . . 170
16.19.1.56 sse_dp_2d() . . . 170
16.19.1.57 sse_dp_3d() . . . 171
16.19.1.58 sse_sp_1d() . . . 172
16.19.1.59 sse_sp_2d() . . . 172
16.19.1.60 sse_sp_3d() . . . 173
16.19.1.61 wnse_dp_1d() . . . 174
16.19.1.62 wnse_dp_2d() . . . 174
16.19.1.63 wnse_dp_3d() . . . 174
16.19.1.64 wnse_sp_1d() . . . 174
16.19.1.65 wnse_sp_2d() . . . 174
16.19.1.66 wnse_sp_3d() . . . 175
16.20 mo_file Module Reference . . . 175
16.20.1 Detailed Description . . . 175
16.20.2 Variable Documentation . . . 175
16.20.2.1 file_defoutput . . . 176
16.20.2.2 file_main . . . 176
16.20.2.3 file_namelist_mhm . . . 176
16.20.2.4 file_namelist_mhm_param . . . 176
16.20.2.5 udefoutput . . . 176
16.20.2.6 unamelist_mhm . . . 176
16.20.2.7 unamelist_mhm_param . . . 176
16.20.2.8 utws . . . 177
16.20.2.9 version . . . 177
16.20.2.10 version_date . . . 177
16.21 mo_finish Module Reference . . . 177
16.21.1 Detailed Description . . . 177

16.21.2 Function/Subroutine Documentation . . . 177
16.21.2.1 finish() . . . 178
16.22mo_global_variables Module Reference . . . 179
16.22.1 Detailed Description . . . 180
16.22.2 Variable Documentation . . . 180
16.22.2.1 basin_avg_tws_obs . . . 181
16.22.2.2 basin_avg_tws_sim . . . 181
16.22.2.3 dirabsvappressure . . . 181
16.22.2.4 direvapotranspiration . . . 181
16.22.2.5 dirmaxtemperature . . . 181
16.22.2.6 dirmintemperature . . . 181
16.22.2.7 dirnetradiation . . . 181
16.22.2.8 dirneutrons . . . 182
16.22.2.9 dirprecipitation . . . 182
16.22.2.10dirreferenceet . . . 182
16.22.2.11dirsoil_moisture . . . 182
16.22.2.12dirtemperature . . . 182
16.22.2.13dirwindspeed . . . 182
16.22.2.14evap_coeff . . . 182
16.22.2.15fday_pet . . . 183
16.22.2.16fday_prec . . . 183
16.22.2.17fday_temp . . . 183
16.22.2.18filetws . . . 183
16.22.2.19fnight_pet . . . 183
16.22.2.20fnight_prec . . . 183
16.22.2.21fnight_temp . . . 183
16.22.2.22inputformat_meteo_forcings . . . 183
16.22.2.23l1_absvappress . . . 184
16.22.2.24l1_aetcanopy . . . 184
16.22.2.25l1_aetsealed . . . 184
16.22.2.26l1_aetsoil . . . 184
16.22.2.27l1_baseflow . . . 184
16.22.2.28l1_et . . . 184
16.22.2.29l1_et_mask . . . 185
16.22.2.30l1_fastrunoff . . . 185
16.22.2.31l1_infilsoil . . . 185
16.22.2.32l1_inter . . . 185
16.22.2.33l1_melt . . . 185
16.22.2.34l1_netrad . . . 185
16.22.2.35l1_neutrons . . . 185

16.22.2.36 l1_neutronsdata . . . 186
16.22.2.37 l1_neutronsdata_mask . . . 186
16.22.2.38 l1_percol . . . 186
16.22.2.39 l1_pet . . . 186
16.22.2.40 l1_pet_calc . . . 186
16.22.2.41 l1_pet_weights . . . 186
16.22.2.42 l1_pre . . . 186
16.22.2.43 l1_pre_weights . . . 187
16.22.2.44 l1_preeffect . . . 187
16.22.2.45 l1_rain . . . 187
16.22.2.46 l1_runoffseal . . . 187
16.22.2.47 l1_satstw . . . 187
16.22.2.48 l1_sealstw . . . 187
16.22.2.49 l1_slowrunoff . . . 187
16.22.2.50 l1_sm . . . 188
16.22.2.51 l1_sm_mask . . . 188
16.22.2.52 l1_snow . . . 188
16.22.2.53 l1_snowpack . . . 188
16.22.2.54 l1_soilmoist . . . 188
16.22.2.55 l1_temp . . . 188
16.22.2.56 l1_temp_weights . . . 188
16.22.2.57 l1_throughfall . . . 189
16.22.2.58 l1_tmax . . . 189
16.22.2.59 l1_tmin . . . 189
16.22.2.60 l1_total_runoff . . . 189
16.22.2.61 l1_unsatstw . . . 189
16.22.2.62 l1_windspeed . . . 189
16.22.2.63 level2 . . . 189
16.22.2.64 neutron_integral_afast . . . 190
16.22.2.65 nmeasperday_tws . . . 190
16.22.2.66 nsoilhorizons_sm_input . . . 190
16.22.2.67 ntimesteps_l1_et . . . 190
16.22.2.68 ntimesteps_l1_neutrons . . . 190
16.22.2.69 ntimesteps_l1_sm . . . 190
16.22.2.70 outputflxstate . . . 190
16.22.2.71 read_meteo_weights . . . 191
16.22.2.72 routingstates . . . 191
16.22.2.73 timestep_et_input . . . 191
16.22.2.74 timestep_model_inputs . . . 191
16.22.2.75 timestep_model_outputs . . . 191

16.22.2.76 timestep_neutrons_input . . . 191
16.22.2.77 timestep_sm_input . . . 191
16.23 mo_grid Module Reference . . . 192
16.23.1 Detailed Description . . . 192
16.23.2 Function/Subroutine Documentation . . . 192
16.23.2.1 calculate_grid_properties() . . . 192
16.23.2.2 geocoordinates() . . . 194
16.23.2.3 init_lowres_level() . . . 194
16.23.2.4 l0_grid_setup() . . . 196
16.23.2.5 mapcoordinates() . . . 196
16.23.2.6 set_basin_indices() . . . 197
16.24 mo_init_states Module Reference . . . 198
16.24.1 Detailed Description . . . 198
16.24.2 Function/Subroutine Documentation . . . 198
16.24.2.1 variables_alloc() . . . 198
16.24.2.2 variables_default_init() . . . 200
16.25 mo_julian Module Reference . . . 201
16.25.1 Detailed Description . . . 202
16.25.2 Function/Subroutine Documentation . . . 203
16.25.2.1 caldat() . . . 203
16.25.2.2 caldat360() . . . 205
16.25.2.3 caldat365() . . . 206
16.25.2.4 caldatjulian() . . . 207
16.25.2.5 date2dec() . . . 208
16.25.2.6 date2dec360() . . . 210
16.25.2.7 date2dec365() . . . 210
16.25.2.8 date2decjulian() . . . 211
16.25.2.9 dec2date() . . . 213
16.25.2.10 dec2date360() . . . 214
16.25.2.11 dec2date365() . . . 216
16.25.2.12 dec2datejulian() . . . 217
16.25.2.13 julday() . . . 219
16.25.2.14 julday360() . . . 220
16.25.2.15 julday365() . . . 221
16.25.2.16 juldayjulian() . . . 223
16.25.2.17 ndays() . . . 225
16.25.2.18 ndyin() . . . 225
16.25.2.19 selectcalendar() . . . 226
16.25.2.20 setcalendarinteger() . . . 227
16.25.2.21 setcalendarstring() . . . 228

16.25.3 Variable Documentation . . . 229
16.25.3.1 calendar . . . 229
16.26 mo_kind Module Reference . . . 229
16.26.1 Detailed Description . . . 229
16.26.2 Variable Documentation . . . 230
16.26.2.1 dp . . . 230
16.26.2.2 dpc . . . 230
16.26.2.3 i1 . . . 230
16.26.2.4 i2 . . . 231
16.26.2.5 i4 . . . 231
16.26.2.6 i8 . . . 231
16.26.2.7 lgt . . . 231
16.26.2.8 sp . . . 232
16.26.2.9 spc . . . 232
16.27 mo_linfit Module Reference . . . 232
16.27.1 Detailed Description . . . 232
16.27.2 Function/Subroutine Documentation . . . 232
16.27.2.1 linfit_dp() . . . 232
16.27.2.2 linfit_sp() . . . 233
16.28 mo_mad Module Reference . . . 233
16.28.1 Function/Subroutine Documentation . . . 233
16.28.1.1 mad_dp() . . . 233
16.28.1.2 mad_sp() . . . 233
16.28.1.3 mad_val_dp() . . . 234
16.28.1.4 mad_val_sp() . . . 234
16.29 mo_mcmc Module Reference . . . 234
16.29.1 Detailed Description . . . 235
16.29.2 Function/Subroutine Documentation . . . 235
16.29.2.1 generatenewparameterset_dp() . . . 235
16.29.2.2 mcmc_dp() . . . 236
16.29.2.3 mcmc_stddev_dp() . . . 236
16.29.2.4 pargen_dp() . . . 237
16.29.2.5 pargennorm_dp() . . . 237
16.30 mo_message Module Reference . . . 237
16.30.1 Detailed Description . . . 238
16.30.2 Function/Subroutine Documentation . . . 238
16.30.2.1 message() . . . 238
16.30.3 Variable Documentation . . . 240
16.30.3.1 message_text . . . 240
16.31 mo_meteo_forcings Module Reference . . . 240

16.31.1 Detailed Description . . . 240
16.31.2 Function/Subroutine Documentation . . . 240
16.31.2.1 chunk_config() . . . 241
16.31.2.2 chunk_size() . . . 242
16.31.2.3 is_read() . . . 243
16.31.2.4 meteo_forcings_wrapper() . . . 244
16.31.2.5 meteo_weights_wrapper() . . . 245
16.31.2.6 prepare_meteo_forcings_data() . . . 247
16.32 mo_mhm Module Reference . . . 248
16.32.1 Detailed Description . . . 248
16.32.2 Function/Subroutine Documentation . . . 249
16.32.2.1 mhm() . . . 249
16.33 mo_mhm_constants Module Reference . . . 254
16.33.1 Detailed Description . . . 255
16.33.2 Variable Documentation . . . 255
16.33.2.1 c1_initstatesm . . . 255
16.33.2.2 cosmic_alpha . . . 255
16.33.2.3 cosmic_bd . . . 255
16.33.2.4 cosmic_l1 . . . 255
16.33.2.5 cosmic_l2 . . . 255
16.33.2.6 cosmic_l3 . . . 256
16.33.2.7 cosmic_l4 . . . 256
16.33.2.8 cosmic_n . . . 256
16.33.2.9 cosmic_vwclat . . . 256
16.33.2.10 desilets_a0 . . . 256
16.33.2.11 desilets_a1 . . . 256
16.33.2.12 desilets_a2 . . . 256
16.33.2.13 duffiedelta1 . . . 256
16.33.2.14 duffiedelta2 . . . 257
16.33.2.15 duffiedr . . . 257
16.33.2.16 h2odens . . . 257
16.33.2.17 harsamconst . . . 257
16.33.2.18 noutflxstate . . . 257
16.33.2.19 p2_initstatefluxes . . . 257
16.33.2.20 p3_initstatefluxes . . . 257
16.33.2.21 p4_initstatefluxes . . . 257
16.33.2.22 p5_initstatefluxes . . . 258
16.33.2.23 satpressureslope1 . . . 258
16.33.2.24 stboltzmann . . . 258
16.33.2.25 tetens_c1 . . . 258

16.33.2.26 tetens_c2 . . . 258
16.33.2.27 tetens_c3 . . . 258
16.34 mo_mhm_eval Module Reference . . . 258
16.34.1 Detailed Description . . . 259
16.34.2 Function/Subroutine Documentation . . . 259
16.34.2.1 mhm_eval() . . . 259
16.35 mo_mhm_read_config Module Reference . . . 262
16.35.1 Detailed Description . . . 262
16.35.2 Function/Subroutine Documentation . . . 262
16.35.2.1 mhm_read_config() . . . 262
16.36 mo_moment Module Reference . . . 265
16.36.1 Function/Subroutine Documentation . . . 266
16.36.1.1 absdev_dp() . . . 266
16.36.1.2 absdev_sp() . . . 266
16.36.1.3 average_dp() . . . 266
16.36.1.4 average_sp() . . . 266
16.36.1.5 central_moment_dp() . . . 266
16.36.1.6 central_moment_sp() . . . 267
16.36.1.7 central_moment_var_dp() . . . 267
16.36.1.8 central_moment_var_sp() . . . 267
16.36.1.9 correlation_dp() . . . 267
16.36.1.10 correlation_sp() . . . 267
16.36.1.11 covariance_dp() . . . 267
16.36.1.12 covariance_sp() . . . 267
16.36.1.13 kurtosis_dp() . . . 268
16.36.1.14 kurtosis_sp() . . . 268
16.36.1.15 mean_dp() . . . 268
16.36.1.16 mean_sp() . . . 268
16.36.1.17 mixed_central_moment_dp() . . . 268
16.36.1.18 mixed_central_moment_sp() . . . 268
16.36.1.19 mixed_central_moment_var_dp() . . . 269
16.36.1.20 mixed_central_moment_var_sp() . . . 269
16.36.1.21 moment_dp() . . . 269
16.36.1.22 moment_sp() . . . 269
16.36.1.23 skewness_dp() . . . 269
16.36.1.24 skewness_sp() . . . 270
16.36.1.25 stddev_dp() . . . 270
16.36.1.26 stddev_sp() . . . 270
16.36.1.27 variance_dp() . . . 270
16.36.1.28 variance_sp() . . . 270

16.37 mo_mpr_constants Module Reference . . . 270
16.37.1 Detailed Description . . . 271
16.37.2 Variable Documentation . . . 272
16.37.2.1 bulkdens_orgmatter . . . 272
16.37.2.2 c1_initstatesm . . . 272
16.37.2.3 field_cap_c1 . . . 272
16.37.2.4 field_cap_c2 . . . 272
16.37.2.5 karman . . . 272
16.37.2.6 ks_c . . . 272
16.37.2.7 lai_factor_surfresi . . . 272
16.37.2.8 lai_offset_surfresi . . . 273
16.37.2.9 max_surfresist . . . 273
16.37.2.10 maxgeounit . . . 273
16.37.2.11 maxnosoilhorizons . . . 273
16.37.2.12 nlcover_class . . . 273
16.37.2.13 p2_initstatefluxes . . . 273
16.37.2.14 p3_initstatefluxes . . . 273
16.37.2.15 p4_initstatefluxes . . . 274
16.37.2.16 p5_initstatefluxes . . . 274
16.37.2.17 pwp_c . . . 274
16.37.2.18 pwp_matpot_thetar . . . 274
16.37.2.19 vgenuchten_sandtresh . . . 274
16.37.2.20 vgenuchtenn_c1 . . . 274
16.37.2.21 vgenuchtenn_c10 . . . 274
16.37.2.22 vgenuchtenn_c11 . . . 274
16.37.2.23 vgenuchtenn_c12 . . . 275
16.37.2.24 vgenuchtenn_c13 . . . 275
16.37.2.25 vgenuchtenn_c14 . . . 275
16.37.2.26 vgenuchtenn_c15 . . . 275
16.37.2.27 vgenuchtenn_c16 . . . 275
16.37.2.28 vgenuchtenn_c17 . . . 275
16.37.2.29 vgenuchtenn_c18 . . . 275
16.37.2.30 vgenuchtenn_c2 . . . 276
16.37.2.31 vgenuchtenn_c3 . . . 276
16.37.2.32 vgenuchtenn_c4 . . . 276
16.37.2.33 vgenuchtenn_c5 . . . 276
16.37.2.34 vgenuchtenn_c6 . . . 276
16.37.2.35 vgenuchtenn_c7 . . . 276
16.37.2.36 vgenuchtenn_c8 . . . 276
16.37.2.37 vgenuchtenn_c9 . . . 276

16.37.2.38 windmeasheight . . . 277
16.38 mo_mpr_eval Module Reference . . . 277
16.38.1 Detailed Description . . . 277
16.38.2 Function/Subroutine Documentation . . . 277
16.38.2.1 mpr_eval() . . . 277
16.39 mo_mpr_file Module Reference . . . 280
16.39.1 Detailed Description . . . 281
16.39.2 Variable Documentation . . . 281
16.39.2.1 file_aspect . . . 281
16.39.2.2 file_geolut . . . 281
16.39.2.3 file_hydrogeoclass . . . 281
16.39.2.4 file_laiclass . . . 282
16.39.2.5 file_lailut . . . 282
16.39.2.6 file_main . . . 282
16.39.2.7 file_meteo_binary_end . . . 282
16.39.2.8 file_meteo_header . . . 282
16.39.2.9 file_namelist_mpr . . . 282
16.39.2.10 file_namelist_mpr_param . . . 282
16.39.2.11 file_slope . . . 283
16.39.2.12 file_soil_database . . . 283
16.39.2.13 file_soil_database_1 . . . 283
16.39.2.14 file_soilclass . . . 283
16.39.2.15 uaspect . . . 283
16.39.2.16 ugeolut . . . 283
16.39.2.17 uhydrogeoclass . . . 284
16.39.2.18 ulaiclass . . . 284
16.39.2.19 ulailut . . . 284
16.39.2.20 umeteo . . . 284
16.39.2.21 umeteo_header . . . 284
16.39.2.22 unamelist_mpr . . . 284
16.39.2.23 unamelist_mpr_param . . . 284
16.39.2.24 uslope . . . 285
16.39.2.25 usoil_database . . . 285
16.39.2.26 usoilclass . . . 285
16.39.2.27 version . . . 285
16.39.2.28 version_date . . . 285
16.40 mo_mpr_global_variables Module Reference . . . 285
16.40.1 Detailed Description . . . 287
16.40.2 Variable Documentation . . . 287
16.40.2.1 dirgridded_lai . . . 287

16.40.2.2 fracsealed_cityarea . . . 287
16.40.2.3 geounitkar . . . 287
16.40.2.4 geounitlist . . . 287
16.40.2.5 horizondepth_mhm . . . 287
16.40.2.6 iflag_soildb . . . 288
16.40.2.7 inputformat_gridded_lai . . . 288
16.40.2.8 l0_asp . . . 288
16.40.2.9 l0_geounit . . . 288
16.40.2.10 l0_gridded_lai . . . 288
16.40.2.11 l0_slope . . . 288
16.40.2.12 l0_slope_emp . . . 288
16.40.2.13 l0_soilid . . . 289
16.40.2.14 l1_aeroresist . . . 289
16.40.2.15 l1_alpha . . . 289
16.40.2.16 l1_degday . . . 289
16.40.2.17 l1_degdayinc . . . 289
16.40.2.18 l1_degdaymax . . . 289
16.40.2.19 l1_degdaynopre . . . 289
16.40.2.20 l1_fasp . . . 290
16.40.2.21 l1_froots . . . 290
16.40.2.22 l1_fsealed . . . 290
16.40.2.23 l1_harsamcoeff . . . 290
16.40.2.24 l1_jarvis_thresh_c1 . . . 290
16.40.2.25 l1_karstloss . . . 290
16.40.2.26 l1_kbaseflow . . . 291
16.40.2.27 l1_kfastflow . . . 291
16.40.2.28 l1_kperco . . . 291
16.40.2.29 l1_kslowflow . . . 291
16.40.2.30 l1_maxinter . . . 291
16.40.2.31 l1_petlaicorfactor . . . 291
16.40.2.32 l1_prietayalpha . . . 291
16.40.2.33 l1_sealedthresh . . . 292
16.40.2.34 l1_soilmoistexp . . . 292
16.40.2.35 l1_soilmoistfc . . . 292
16.40.2.36 l1_soilmoistsat . . . 292
16.40.2.37 l1_surfresist . . . 292
16.40.2.38 l1_tempthresh . . . 292
16.40.2.39 l1_unsatthresh . . . 292
16.40.2.40 l1_wiltingpoint . . . 293
16.40.2.41 lailut . . . 293

16.40.2.42 laiper . . . 293
16.40.2.43 laiunitlist . . . 293
16.40.2.44 ngeounits . . . 293
16.40.2.45 nlai . . . 293
16.40.2.46 nlaiclass . . . 293
16.40.2.47 nsoilhorizons_mhm . . . 294
16.40.2.48 nsoiltypes . . . 294
16.40.2.49 soildb . . . 294
16.40.2.50 tillagedepth . . . 294
16.40.2.51 timestep_lai_input . . . 294
16.41 mo_mpr_pet Module Reference . . . 294
16.41.1 Detailed Description . . . 295
16.41.2 Function/Subroutine Documentation . . . 295
16.41.2.1 bulksurface_resistance() . . . 295
16.41.2.2 pet_correctbyasp() . . . 296
16.41.2.3 pet_correctbylai() . . . 297
16.41.2.4 priestley_taylor_alpha() . . . 299
16.42 mo_mpr_read_config Module Reference . . . 300
16.42.1 Detailed Description . . . 300
16.42.2 Function/Subroutine Documentation . . . 300
16.42.2.1 mpr_read_config() . . . 301
16.43 mo_mpr_restart Module Reference . . . 302
16.43.1 Detailed Description . . . 303
16.43.2 Function/Subroutine Documentation . . . 303
16.43.2.1 unpack_field_and_write_1d_dp() . . . 303
16.43.2.2 unpack_field_and_write_1d_i4() . . . 303
16.43.2.3 unpack_field_and_write_2d_dp() . . . 304
16.43.2.4 unpack_field_and_write_3d_dp() . . . 304
16.43.2.5 write_eff_params() . . . 304
16.43.2.6 write_mpr_restart_files() . . . 305
16.44 mo_mpr_runoff Module Reference . . . 306
16.44.1 Detailed Description . . . 307
16.44.2 Function/Subroutine Documentation . . . 307
16.44.2.1 mpr_runoff() . . . 307
16.45 mo_mpr_smhorizons Module Reference . . . 308
16.45.1 Detailed Description . . . 309
16.45.2 Function/Subroutine Documentation . . . 309
16.45.2.1 mpr_smhorizons() . . . 309
16.46 mo_mpr_soilmoist Module Reference . . . 311
16.46.1 Detailed Description . . . 312

16.46.2 Function/Subroutine Documentation . . . 312
16.46.2.1 field_cap() . . . 312
16.46.2.2 genuchten() . . . 313
16.46.2.3 hydro_cond() . . . 314
16.46.2.4 mpr_sm() . . . 315
16.46.2.5 pwp() . . . 317
16.47 mo_mpr_startup Module Reference . . . 318
16.47.1 Detailed Description . . . 318
16.47.2 Function/Subroutine Documentation . . . 319
16.47.2.1 init_eff_params() . . . 319
16.47.2.2 l0_check_input() . . . 320
16.47.2.3 l0_variable_init() . . . 321
16.47.2.4 mpr_initialize() . . . 322
16.48 mo_mrm_constants Module Reference . . . 323
16.48.1 Detailed Description . . . 324
16.48.2 Variable Documentation . . . 324
16.48.2.1 deltah . . . 324
16.48.2.2 given_ts . . . 324
16.48.2.3 maxnogauges . . . 324
16.48.2.4 noutflxstate . . . 324
16.48.2.5 nroutingstates . . . 325
16.48.2.6 rout_space_weight . . . 325
16.49 mo_mrm_eval Module Reference . . . 325
16.49.1 Detailed Description . . . 325
16.49.2 Function/Subroutine Documentation . . . 325
16.49.2.1 mrm_eval() . . . 325
16.50 mo_mrm_file Module Reference . . . 327
16.50.1 Detailed Description . . . 329
16.50.2 Variable Documentation . . . 329
16.50.2.1 file_config . . . 329
16.50.2.2 file_daily_discharge . . . 329
16.50.2.3 file_defoutput . . . 329
16.50.2.4 file_facc . . . 329
16.50.2.5 file_fdir . . . 329
16.50.2.6 file_gaugeloc . . . 330
16.50.2.7 file_gw_output . . . 330
16.50.2.8 file_main . . . 330
16.50.2.9 file_mrm_output . . . 330
16.50.2.10 file_namelist_mrm . . . 330
16.50.2.11 file_namelist_param_mrm . . . 330

16.50.2.12file_slope . . . 331
16.50.2.13ncfile_discharge . . . 331
16.50.2.14uconfig . . . 331
16.50.2.15udaily_discharge . . . 331
16.50.2.16udefoutput . . . 331
16.50.2.17udischarge . . . 331
16.50.2.18ufacc . . . 331
16.50.2.19ufdir . . . 332
16.50.2.20ugaugeloc . . . 332
16.50.2.21unamelist_mrm . . . 332
16.50.2.22unamelist_param_mrm . . . 332
16.50.2.23uslope . . . 332
16.50.2.24version . . . 332
16.50.2.25version_date . . . 333
16.51 mo_mrm_global_variables Module Reference . . . 333
16.51.1 Detailed Description . . . 334
16.51.2 Variable Documentation . . . 334
16.51.2.1 basin_mrm . . . 335
16.51.2.2 dirbankfullrunoff . . . 335
16.51.2.3 dirgauges . . . 335
16.51.2.4 dirtotalrunoff . . . 335
16.51.2.5 filenametotalrunoff . . . 335
16.51.2.6 gauge . . . 335
16.51.2.7 gw_coupling . . . 336
16.51.2.8 inflowgauge . . . 336
16.51.2.9 is_start . . . 336
16.51.2.10l0_celerity . . . 336
16.51.2.11l0_channel_depth . . . 336
16.51.2.12l0_channel_elevation . . . 336
16.51.2.13l0_dracell . . . 336
16.51.2.14l0_drasc . . . 337
16.51.2.15l0_facc . . . 337
16.51.2.16l0_fdir . . . 337
16.51.2.17l0_floodplain . . . 337
16.51.2.18l0_gaugeloc . . . 337
16.51.2.19l0_inflowgaugeloc . . . 337
16.51.2.20l0_l11_remap . . . 338
16.51.2.21l0_noutlet . . . 338
16.51.2.22l0_river_head_mon_sum . . . 338
16.51.2.23l0_slope . . . 338

16.51.2.240_streamnet . . . 338
16.51.2.2511_afloodplain . . . 338
16.51.2.2611_areacell . . . 338
16.51.2.2711_bankfull_runoff_in . . . 338
16.51.2.2811_c1 . . . 339
16.51.2.2911_c2 . . . 339
16.51.2.3011_celerity . . . 339
16.51.2.3111_cellcoor . . . 339
16.51.2.3211_colout . . . 339
16.51.2.3311_facc . . . 339
16.51.2.3411_fcol . . . 339
16.51.2.3511_fdir . . . 340
16.51.2.3611_fromn . . . 340
16.51.2.3711_frow . . . 340
16.51.2.3811_k . . . 340
16.51.2.3911_l1_id . . . 340
16.51.2.4011_label . . . 340
16.51.2.4111_length . . . 341
16.51.2.4211_linkin_facc . . . 341
16.51.2.4311_meandering . . . 341
16.51.2.4411_netperm . . . 341
16.51.2.4511_nlinkfracfpimp . . . 341
16.51.2.4611_noutlets . . . 341
16.51.2.4711_qmod . . . 341
16.51.2.4811_qout . . . 342
16.51.2.4911_qtin . . . 342
16.51.2.5011_qtr . . . 342
16.51.2.5111_rorder . . . 342
16.51.2.5211_rowout . . . 342
16.51.2.5311_sink . . . 342
16.51.2.5411_slope . . . 343
16.51.2.5511_tcol . . . 343
16.51.2.5611_ton . . . 343
16.51.2.5711_trow . . . 343
16.51.2.5811_tsrout . . . 343
16.51.2.5911_xi . . . 343
16.51.2.601_l11_id . . . 344
16.51.2.611_l11_remap . . . 344
16.51.2.621_total_runoff_in . . . 344
16.51.2.63evel11 . . . 344

16.51.2.64 mrm_runoff . . . 344
16.51.2.65 ngaugestotal . . . 344
16.51.2.66 ninflowgaugestotal . . . 345
16.51.2.67 nmeasperday . . . 345
16.51.2.68 ntstepday . . . 345
16.51.2.69 outputflxstate_mrm . . . 345
16.51.2.70 timestep_model_outputs_mrm . . . 345
16.51.2.71 varnametotalrunoff . . . 345
16.52 mo_mrm_init Module Reference . . . 345
16.52.1 Detailed Description . . . 346
16.52.2 Function/Subroutine Documentation . . . 346
16.52.2.1 config_output() . . . 346
16.52.2.2 l0_check_input_routing() . . . 347
16.52.2.3 mrm_init() . . . 348
16.52.2.4 print_startup_message() . . . 351
16.52.2.5 variables_alloc_routing() . . . 352
16.52.2.6 variables_default_init_routing() . . . 353
16.53 mo_mrm_mpr Module Reference . . . 353
16.53.1 Detailed Description . . . 354
16.53.2 Function/Subroutine Documentation . . . 354
16.53.2.1 mrm_init_param() . . . 354
16.53.2.2 mrm_update_param() . . . 355
16.53.2.3 reg_rout() . . . 356
16.54 mo_mrm_net_startup Module Reference . . . 358
16.54.1 Detailed Description . . . 359
16.54.2 Function/Subroutine Documentation . . . 359
16.54.2.1 celllength() . . . 359
16.54.2.2 get_distance_two_lat_lon_points() . . . 360
16.54.2.3 l11_calc_celerity() . . . 361
16.54.2.4 l11_flow_accumulation() . . . 362
16.54.2.5 l11_flow_direction() . . . 363
16.54.2.6 l11_fraction_sealed_floodplain() . . . 365
16.54.2.7 l11_l1_mapping() . . . 366
16.54.2.8 l11_link_location() . . . 367
16.54.2.9 l11_routing_order() . . . 368
16.54.2.10 l11_set_drain_outlet_gauges() . . . 369
16.54.2.11 l11_set_network_topology() . . . 370
16.54.2.12 l11_stream_features() . . . 371
16.54.2.13 movedownonecell() . . . 372
16.54.2.14 moveup() . . . 373

16.55mo_mrm_objective_function_runoff Module Reference . . . 374
16.55.1 Detailed Description . . . 375
16.55.2 Function/Subroutine Documentation . . . 375
16.55.2.1 extract_runoff() . . . 375
16.55.2.2 loglikelihood_evin2013_2() . . . 377
16.55.2.3 loglikelihood_stddev() . . . 379
16.55.2.4 loglikelihood_trend_no_autocorr() . . . 380
16.55.2.5 multi_objective_ae_fdc_lsv_nse_djf() . . . 381
16.55.2.6 multi_objective_lnnse_highflow_lnnse_lowflow() . . . 382
16.55.2.7 multi_objective_lnnse_highflow_lnnse_lowflow_2() . . . 384
16.55.2.8 multi_objective_nse_lnnse() . . . 385
16.55.2.9 multi_objective_runoff() . . . 386
16.55.2.10objective_equal_nse_lnnse() . . . 387
16.55.2.11objective_kge() . . . 389
16.55.2.12objective_lnnse() . . . 391
16.55.2.13objective_multiple_gauges_kge_power6() . . . 392
16.55.2.14objective_nse() . . . 393
16.55.2.15objective_power6_nse_lnnse() . . . 394
16.55.2.16objective_sse() . . . 396
16.55.2.17objective_weighted_nse() . . . 397
16.55.2.18parameter_regularization() . . . 398
16.55.2.19single_objective_runoff() . . . 399
16.56mo_mrm_read_config Module Reference . . . 400
16.56.1 Detailed Description . . . 401
16.56.2 Function/Subroutine Documentation . . . 401
16.56.2.1 mrm_read_config() . . . 401
16.56.2.2 read_mrm_routing_params() . . . 402
16.57mo_mrm_read_data Module Reference . . . 404
16.57.1 Detailed Description . . . 404
16.57.2 Function/Subroutine Documentation . . . 404
16.57.2.1 mrm_read_bankfull_runoff() . . . 404
16.57.2.2 mrm_read_discharge() . . . 405
16.57.2.3 mrm_read_l0_data() . . . 406
16.57.2.4 mrm_read_total_runoff() . . . 407
16.57.2.5 rotate_fdir_variable() . . . 408
16.58mo_mrm_restart Module Reference . . . 409
16.58.1 Detailed Description . . . 409
16.58.2 Function/Subroutine Documentation . . . 410
16.58.2.1 mrm_read_restart_config() . . . 410
16.58.2.2 mrm_read_restart_states() . . . 411

16.58.2.3 mrm_write_restart() . . . 412
16.59 mo_mrm_river_head Module Reference . . . 413
16.59.1 Function/Subroutine Documentation . . . 414
16.59.1.1 avg_and_write_timestep() . . . 414
16.59.1.2 calc_channel_elevation() . . . 414
16.59.1.3 calc_river_head() . . . 415
16.59.1.4 calc_slope() . . . 416
16.59.1.5 create_output() . . . 417
16.59.1.6 init_masked_zeros_l0() . . . 417
16.59.1.7 reset_sum() . . . 418
16.59.2 Variable Documentation . . . 418
16.59.2.1 nc_riverhead . . . 419
16.59.2.2 nc_time . . . 419
16.59.2.3 sum_counter . . . 419
16.59.2.4 time_counter . . . 419
16.60 mo_mrm_routing Module Reference . . . 419
16.60.1 Detailed Description . . . 419
16.60.2 Function/Subroutine Documentation . . . 420
16.60.2.1 add_inflow() . . . 420
16.60.2.2 l11_routing() . . . 421
16.60.2.3 l11_runoff_acc() . . . 423
16.60.2.4 mrm_routing() . . . 424
16.61 mo_mrm_signatures Module Reference . . . 427
16.61.1 Detailed Description . . . 427
16.61.2 Function/Subroutine Documentation . . . 428
16.61.2.1 autocorrelation() . . . 428
16.61.2.2 flowdurationcurve() . . . 428
16.61.2.3 limb_densities() . . . 430
16.61.2.4 maximummonthlyflow() . . . 431
16.61.2.5 moments() . . . 432
16.61.2.6 peakdistribution() . . . 434
16.61.2.7 runoffratio() . . . 434
16.61.2.8 zeroflowratio() . . . 436
16.62 mo_mrm_write Module Reference . . . 436
16.62.1 Detailed Description . . . 437
16.62.2 Function/Subroutine Documentation . . . 437
16.62.2.1 mrm_write() . . . 437
16.62.2.2 mrm_write_optifile() . . . 438
16.62.2.3 mrm_write_optinamelist() . . . 439
16.62.2.4 mrm_write_output_fluxes() . . . 441

16.62.2.5 write_configfile() . . . 443
16.62.2.6 write_daily_obs_sim_discharge() . . . 444
16.62.3 Variable Documentation . . . 445
16.62.3.1 average_counter . . . 446
16.62.3.2 day_counter . . . 446
16.62.3.3 month_counter . . . 446
16.62.3.4 nc . . . 446
16.62.3.5 year_counter . . . 446
16.63 mo_mrm_write_fluxes_states Module Reference . . . 446
16.63.1 Detailed Description . . . 447
16.63.2 Function/Subroutine Documentation . . . 447
16.63.2.1 close() . . . 447
16.63.2.2 createoutputfile() . . . 448
16.63.2.3 newoutputdataset() . . . 449
16.63.2.4 newoutputvariable() . . . 450
16.63.2.5 updatedataset() . . . 451
16.63.2.6 updatevariable() . . . 452
16.63.2.7 writetimestep() . . . 453
16.63.2.8 writevariableattributes() . . . 453
16.63.2.9 writevariabletimestep() . . . 454
16.64 mo_multi_param_reg Module Reference . . . 455
16.64.1 Detailed Description . . . 455
16.64.2 Function/Subroutine Documentation . . . 455
16.64.2.1 aerodynamical_resistance() . . . 456
16.64.2.2 baseflow_param() . . . 457
16.64.2.3 canopy_intercept_param() . . . 458
16.64.2.4 iper_thres_runoff() . . . 459
16.64.2.5 karstic_layer() . . . 460
16.64.2.6 mpr() . . . 462
16.64.2.7 snow_acc_melt_param() . . . 466
16.65 mo_ncread Module Reference . . . 467
16.65.1 Function/Subroutine Documentation . . . 468
16.65.1.1 check() . . . 468
16.65.1.2 get_info() . . . 469
16.65.1.3 get_ncdim() . . . 471
16.65.1.4 get_ncdimatt() . . . 473
16.65.1.5 get_ncvar_0d_dp() . . . 474
16.65.1.6 get_ncvar_0d_i1() . . . 474
16.65.1.7 get_ncvar_0d_i4() . . . 475
16.65.1.8 get_ncvar_0d_sp() . . . 475

16.65.1.9 get_ncvar_1d_dp() . . . 476
16.65.1.10 get_ncvar_1d_i1() . . . 476
16.65.1.11 get_ncvar_1d_i4() . . . 477
16.65.1.12 get_ncvar_1d_sp() . . . 477
16.65.1.13 get_ncvar_2d_dp() . . . 478
16.65.1.14 get_ncvar_2d_i1() . . . 478
16.65.1.15 get_ncvar_2d_i4() . . . 479
16.65.1.16 get_ncvar_2d_sp() . . . 479
16.65.1.17 get_ncvar_3d_dp() . . . 480
16.65.1.18 get_ncvar_3d_i1() . . . 480
16.65.1.19 get_ncvar_3d_i4() . . . 481
16.65.1.20 get_ncvar_3d_sp() . . . 481
16.65.1.21 get_ncvar_4d_dp() . . . 482
16.65.1.22 get_ncvar_4d_i1() . . . 482
16.65.1.23 get_ncvar_4d_i4() . . . 483
16.65.1.24 get_ncvar_4d_sp() . . . 483
16.65.1.25 get_ncvar_5d_dp() . . . 484
16.65.1.26 get_ncvar_5d_i1() . . . 484
16.65.1.27 get_ncvar_5d_i4() . . . 485
16.65.1.28 get_ncvar_5d_sp() . . . 485
16.65.1.29 get_ncvaratt() . . . 486
16.65.1.30 ncclose() . . . 486
16.65.1.31 ncopen() . . . 487
16.66 mo_ncwrite Module Reference . . . 487
16.66.1 Function/Subroutine Documentation . . . 489
16.66.1.1 check() . . . 489
16.66.1.2 close_netcdf() . . . 490
16.66.1.3 create_netcdf() . . . 492
16.66.1.4 dump_netcdf_1d_dp() . . . 493
16.66.1.5 dump_netcdf_1d_i4() . . . 493
16.66.1.6 dump_netcdf_1d_sp() . . . 494
16.66.1.7 dump_netcdf_2d_dp() . . . 494
16.66.1.8 dump_netcdf_2d_i4() . . . 495
16.66.1.9 dump_netcdf_2d_sp() . . . 495
16.66.1.10 dump_netcdf_3d_dp() . . . 496
16.66.1.11 dump_netcdf_3d_i4() . . . 496
16.66.1.12 dump_netcdf_3d_sp() . . . 497
16.66.1.13 dump_netcdf_4d_dp() . . . 497
16.66.1.14 dump_netcdf_4d_i4() . . . 498
16.66.1.15 dump_netcdf_4d_sp() . . . 498

16.66.1.16 dump_netcdf_5d_dp() . . . 499
16.66.1.17 dump_netcdf_5d_i4() . . . 499
16.66.1.18 dump_netcdf_5d_sp() . . . 500
16.66.1.19 open_netcdf() . . . 500
16.66.1.20 var2nc_1d_dp() . . . 502
16.66.1.21 var2nc_1d_i4() . . . 503
16.66.1.22 var2nc_1d_sp() . . . 504
16.66.1.23 var2nc_2d_dp() . . . 504
16.66.1.24 var2nc_2d_i4() . . . 505
16.66.1.25 var2nc_2d_sp() . . . 506
16.66.1.26 var2nc_3d_dp() . . . 506
16.66.1.27 var2nc_3d_i4() . . . 507
16.66.1.28 var2nc_3d_sp() . . . 508
16.66.1.29 var2nc_4d_dp() . . . 508
16.66.1.30 var2nc_4d_i4() . . . 509
16.66.1.31 var2nc_4d_sp() . . . 510
16.66.1.32 var2nc_5d_dp() . . . 510
16.66.1.33 var2nc_5d_i4() . . . 511
16.66.1.34 var2nc_5d_sp() . . . 512
16.66.1.35 write_dynamic_netcdf() . . . 512
16.66.1.36 write_static_netcdf() . . . 513
16.66.2 Variable Documentation . . . 513
16.66.2.1 dnc . . . 513
16.66.2.2 gatt . . . 513
16.66.2.3 maxlen . . . 514
16.66.2.4 nattdim . . . 514
16.66.2.5 ndims . . . 514
16.66.2.6 ngatt . . . 514
16.66.2.7 nmaxatt . . . 514
16.66.2.8 nmaxdim . . . 514
16.66.2.9 nvars . . . 514
16.66.2.10 v . . . 514
16.67 mo_netcdf Module Reference . . . 515
16.67.1 Detailed Description . . . 518
16.67.2 Function/Subroutine Documentation . . . 518
16.67.2.1 check() . . . 518
16.67.2.2 close() . . . 519
16.67.2.3 equalncdimensions() . . . 519
16.67.2.4 getdata1df32() . . . 519
16.67.2.5 getdata1df64() . . . 520

16.67.2.6 getdata1di16() . . . 520
16.67.2.7 getdata1di32() . . . 521
16.67.2.8 getdata1di64() . . . 521
16.67.2.9 getdata1di8() . . . 522
16.67.2.10 getdata2df32() . . . 522
16.67.2.11 getdata2df64() . . . 523
16.67.2.12 getdata2di16() . . . 523
16.67.2.13 getdata2di32() . . . 524
16.67.2.14 getdata2di64() . . . 524
16.67.2.15 getdata2di8() . . . 525
16.67.2.16 getdata3df32() . . . 525
16.67.2.17 getdata3df64() . . . 526
16.67.2.18 getdata3di16() . . . 526
16.67.2.19 getdata3di32() . . . 527
16.67.2.20 getdata3di64() . . . 527
16.67.2.21 getdata3di8() . . . 528
16.67.2.22 getdata4df32() . . . 528
16.67.2.23 getdata4df64() . . . 529
16.67.2.24 getdata4di16() . . . 529
16.67.2.25 getdata4di32() . . . 530
16.67.2.26 getdata4di64() . . . 530
16.67.2.27 getdata4di8() . . . 531
16.67.2.28 getdata5df32() . . . 531
16.67.2.29 getdata5df64() . . . 532
16.67.2.30 getdata5di16() . . . 532
16.67.2.31 getdata5di32() . . . 533
16.67.2.32 getdata5di64() . . . 533
16.67.2.33 getdata5di8() . . . 534
16.67.2.34 getdatascalarf32() . . . 534
16.67.2.35 getdatascalarf64() . . . 534
16.67.2.36 getdatascalari16() . . . 535
16.67.2.37 getdatascalari32() . . . 535
16.67.2.38 getdatascalari64() . . . 536
16.67.2.39 getdatascalari8() . . . 536
16.67.2.40 getdimensionbyid() . . . 537
16.67.2.41 getdimensionbyname() . . . 537
16.67.2.42 getdimensionlength() . . . 538
16.67.2.43 getdimensionname() . . . 538
16.67.2.44 getdtypefrominteger() . . . 538
16.67.2.45 getdtypefromstring() . . . 539

16.67.2.46 getglobalattributechar() . . . 539
16.67.2.47 getglobalattributef32() . . . 540
16.67.2.48 getglobalattributef64() . . . 540
16.67.2.49 getglobalattributei16() . . . 540
16.67.2.50 getglobalattributei32() . . . 541
16.67.2.51 getglobalattributei64() . . . 541
16.67.2.52 getglobalattributei8() . . . 542
16.67.2.53 getnodimensions() . . . 542
16.67.2.54 getnovariables() . . . 542
16.67.2.55 getreaddatashape() . . . 543
16.67.2.56 getunlimiteddimension() . . . 544
16.67.2.57 getvariableattributechar() . . . 545
16.67.2.58 getvariableattributef32() . . . 545
16.67.2.59 getvariableattributef64() . . . 546
16.67.2.60 getvariableattributei16() . . . 546
16.67.2.61 getvariableattributei32() . . . 546
16.67.2.62 getvariableattributei64() . . . 547
16.67.2.63 getvariableattributei8() . . . 547
16.67.2.64 getvariablebyname() . . . 548
16.67.2.65 getvariabledimensions() . . . 548
16.67.2.66 getvariabledtype() . . . 548
16.67.2.67 getvariablefillvaluef32() . . . 549
16.67.2.68 getvariablefillvaluef64() . . . 549
16.67.2.69 getvariablefillvaluei16() . . . 549
16.67.2.70 getvariablefillvaluei32() . . . 549
16.67.2.71 getvariablefillvaluei64() . . . 549
16.67.2.72 getvariablefillvaluei8() . . . 550
16.67.2.73 getvariableids() . . . 550
16.67.2.74 getvariablename() . . . 550
16.67.2.75 getvariables() . . . 550
16.67.2.76 getvariableshape() . . . 551
16.67.2.77 hasattribute() . . . 551
16.67.2.78 hasdimension() . . . 551
16.67.2.79 hasvariable() . . . 551
16.67.2.80 initncdataset() . . . 551
16.67.2.81 initncdimension() . . . 551
16.67.2.82 initncvariable() . . . 552
16.67.2.83 isdatasetunlimited() . . . 552
16.67.2.84 isunlimiteddimension() . . . 552
16.67.2.85 isunlimitedvariable() . . . 552

16.67.2.86 newncdataset() . . . 552
16.67.2.87 newncdimension() . . . 553
16.67.2.88 newncvariable() . . . 553
16.67.2.89 setdata1df32() . . . 553
16.67.2.90 setdata1df64() . . . 553
16.67.2.91 setdata1di16() . . . 554
16.67.2.92 setdata1di32() . . . 554
16.67.2.93 setdata1di64() . . . 555
16.67.2.94 setdata1di8() . . . 555
16.67.2.95 setdata2df32() . . . 556
16.67.2.96 setdata2df64() . . . 556
16.67.2.97 setdata2di16() . . . 557
16.67.2.98 setdata2di32() . . . 557
16.67.2.99 setdata2di64() . . . 558
16.67.2.100 setdata2di8() . . . 558
16.67.2.101 setdata3df32() . . . 559
16.67.2.102 setdata3df64() . . . 559
16.67.2.103 setdata3di16() . . . 560
16.67.2.104 setdata3di32() . . . 560
16.67.2.105 setdata3di64() . . . 561
16.67.2.106 setdata3di8() . . . 561
16.67.2.107 setdata4df32() . . . 562
16.67.2.108 setdata4df64() . . . 562
16.67.2.109 setdata4di16() . . . 563
16.67.2.110 setdata4di32() . . . 563
16.67.2.111 setdata4di64() . . . 564
16.67.2.112 setdata4di8() . . . 564
16.67.2.113 setdata5df32() . . . 565
16.67.2.114 setdata5df64() . . . 565
16.67.2.115 setdata5di16() . . . 566
16.67.2.116 setdata5di32() . . . 566
16.67.2.117 setdata5di64() . . . 567
16.67.2.118 setdata5di8() . . . 567
16.67.2.119 setdatascalarf32() . . . 568
16.67.2.120 setdatascalarf64() . . . 568
16.67.2.121 setdatascalari16() . . . 569
16.67.2.122 setdatascalari32() . . . 569
16.67.2.123 setdatascalari64() . . . 569
16.67.2.124 setdatascalari8() . . . 570
16.67.2.125 setdimension() . . . 570

16.67.2.126 netglobalattributechar() . . . 571
16.67.2.127 netglobalattributef32() . . . 571
16.67.2.128 netglobalattributef64() . . . 571
16.67.2.129 netglobalattributei16() . . . 572
16.67.2.130 netglobalattributei32() . . . 572
16.67.2.131 netglobalattributei64() . . . 573
16.67.2.132 netglobalattributei8() . . . 573
16.67.2.133 netvariableattributechar() . . . 573
16.67.2.134 netvariableattributef32() . . . 574
16.67.2.135 netvariableattributef64() . . . 574
16.67.2.136 netvariableattributei16() . . . 575
16.67.2.137 netvariableattributei32() . . . 575
16.67.2.138 netvariableattributei64() . . . 575
16.67.2.139 netvariableattributei8() . . . 576
16.67.2.140 netvariablefillvaluef32() . . . 576
16.67.2.141 netvariablefillvaluef64() . . . 576
16.67.2.142 netvariablefillvaluei16() . . . 576
16.67.2.143 netvariablefillvaluei32() . . . 577
16.67.2.144 netvariablefillvaluei64() . . . 577
16.67.2.145 netvariablefillvaluei8() . . . 577
16.67.2.146 netvariablewithids() . . . 577
16.67.2.147 netvariablewithnames() . . . 578
16.67.2.148 netvariablewithtypes() . . . 579
16.68 mo_neutrons Module Reference . . . 579
16.68.1 Detailed Description . . . 580
16.68.2 Function/Subroutine Documentation . . . 580
16.68.2.1 approx_mon_int() . . . 580
16.68.2.2 approx_mon_int_eps() . . . 581
16.68.2.3 approx_mon_int_steps() . . . 582
16.68.2.4 cosmic() . . . 583
16.68.2.5 desiletsn0() . . . 584
16.68.2.6 intgrandfast() . . . 585
16.68.2.7 lookupintegral() . . . 586
16.68.2.8 oldintegration() . . . 586
16.68.2.9 tabularintegralafast() . . . 587
16.69 mo_nml Module Reference . . . 588
16.69.1 Detailed Description . . . 589
16.69.2 Function/Subroutine Documentation . . . 589
16.69.2.1 close_nml() . . . 589
16.69.2.2 open_nml() . . . 590

16.69.2.3 position_nml() . . . 591
16.69.3 Variable Documentation . . . 593
16.69.3.1 length_error . . . 593
16.69.3.2 missing . . . 593
16.69.3.3 nunitnml . . . 593
16.69.3.4 positioned . . . 593
16.69.3.5 read_error . . . 594
16.70 mo_objective_function Module Reference . . . 594
16.70.1 Detailed Description . . . 594
16.70.2 Function/Subroutine Documentation . . . 595
16.70.2.1 extract_basin_avg_tws() . . . 595
16.70.2.2 objective() . . . 596
16.70.2.3 objective_et_kge_catchment_avg() . . . 598
16.70.2.4 objective_kge_q_et() . . . 599
16.70.2.5 objective_kge_q_rmse_et() . . . 600
16.70.2.6 objective_kge_q_rmse_tws() . . . 601
16.70.2.7 objective_kge_q_sm_corr() . . . 603
16.70.2.8 objective_neutrons_kge_catchment_avg() . . . 604
16.70.2.9 objective_sm_corr() . . . 605
16.70.2.10 objective_sm_kge_catchment_avg() . . . 607
16.70.2.11 objective_sm_pd() . . . 608
16.70.2.12 objective_sm_sse_standard_score() . . . 609
16.71 mo_optimization Module Reference . . . 610
16.71.1 Detailed Description . . . 611
16.71.2 Function/Subroutine Documentation . . . 611
16.71.2.1 optimization() . . . 611
16.72 mo_optimization_utils Module Reference . . . 612
16.73 mo_orderpack Module Reference . . . 613
16.73.1 Detailed Description . . . 615
16.73.2 Function/Subroutine Documentation . . . 615
16.73.2.1 d_ctrper() . . . 615
16.73.2.2 d_fndnth() . . . 615
16.73.2.3 d_indmed() . . . 615
16.73.2.4 d_indnth() . . . 616
16.73.2.5 d_inspar() . . . 616
16.73.2.6 d_inssor() . . . 616
16.73.2.7 d_med() . . . 616
16.73.2.8 d_median() . . . 617
16.73.2.9 d_mrgref() . . . 617
16.73.2.10 d_mrgrnk() . . . 617

16.73.2.11d_mulcnt() . . . 617
16.73.2.12d_nearless() . . . 617
16.73.2.13d_rapknr() . . . 617
16.73.2.14d_refpar() . . . 618
16.73.2.15d_refsor() . . . 618
16.73.2.16d_rinpar() . . . 618
16.73.2.17d_rnkpar() . . . 618
16.73.2.18d_subsor() . . . 619
16.73.2.19d_uniinv() . . . 619
16.73.2.20d_unipar() . . . 619
16.73.2.21d_unirnk() . . . 619
16.73.2.22d_unista() . . . 619
16.73.2.23d_valmed() . . . 620
16.73.2.24d_valnth() . . . 620
16.73.2.25i_ctrper() . . . 620
16.73.2.26i_fndnth() . . . 620
16.73.2.27i_indmed() . . . 620
16.73.2.28i_indnth() . . . 620
16.73.2.29i_inspar() . . . 621
16.73.2.30i_inssor() . . . 621
16.73.2.31i_med() . . . 621
16.73.2.32i_median() . . . 622
16.73.2.33i_mrgref() . . . 622
16.73.2.34i_mrgrnk() . . . 622
16.73.2.35i_mulcnt() . . . 622
16.73.2.36i_nearless() . . . 622
16.73.2.37i_rapknr() . . . 622
16.73.2.38i_refpar() . . . 622
16.73.2.39i_refsor() . . . 623
16.73.2.40i_rinpar() . . . 623
16.73.2.41i_rnkpar() . . . 623
16.73.2.42i_subsor() . . . 623
16.73.2.43i_uniinv() . . . 624
16.73.2.44i_unipar() . . . 624
16.73.2.45i_unirnk() . . . 624
16.73.2.46i_unista() . . . 624
16.73.2.47i_valmed() . . . 624
16.73.2.48i_valnth() . . . 625
16.73.2.49r_ctrper() . . . 625
16.73.2.50r_fndnth() . . . 625

16.73.2.51 r_indmed() . . . 625
16.73.2.52 r_indnth() . . . 625
16.73.2.53 r_inspar() . . . 625
16.73.2.54 r_inssor() . . . 626
16.73.2.55 r_med() . . . 626
16.73.2.56 r_median() . . . 626
16.73.2.57 r_mrgref() . . . 627
16.73.2.58 r_mrgrnk() . . . 627
16.73.2.59 r_mulcnt() . . . 627
16.73.2.60 r_nearless() . . . 627
16.73.2.61 r_rapknr() . . . 627
16.73.2.62 r_refpar() . . . 627
16.73.2.63 r_refsor() . . . 627
16.73.2.64 r_rinpar() . . . 628
16.73.2.65 r_rnkpar() . . . 628
16.73.2.66 r_subsor() . . . 628
16.73.2.67 r_uniinv() . . . 629
16.73.2.68 r_unipar() . . . 629
16.73.2.69 r_unirnk() . . . 629
16.73.2.70 r_unista() . . . 629
16.73.2.71 r_valmed() . . . 629
16.73.2.72 r_valnth() . . . 629
16.73.2.73 sort_index_dp() . . . 629
16.73.2.74 sort_index_i4() . . . 630
16.73.2.75 sort_index_sp() . . . 630
16.73.3 Variable Documentation . . . 630
16.73.3.1 idont . . . 630
16.74 mo_percentile Module Reference . . . 630
16.74.1 Function/Subroutine Documentation . . . 630
16.74.1.1 median_dp() . . . 631
16.74.1.2 median_sp() . . . 631
16.74.1.3 n_element_dp() . . . 631
16.74.1.4 n_element_sp() . . . 631
16.74.1.5 percentile_0d_dp() . . . 631
16.74.1.6 percentile_0d_sp() . . . 631
16.74.1.7 percentile_1d_dp() . . . 632
16.74.1.8 percentile_1d_sp() . . . 632
16.74.1.9 qmedian_dp() . . . 632
16.74.1.10 qmedian_sp() . . . 632
16.75 mo_pet Module Reference . . . 632

16.75.1 Detailed Description . . . 633
16.75.2 Function/Subroutine Documentation . . . 633
16.75.2.1 extraterr_rad_approx() . . . 633
16.75.2.2 pet_hargreaves() . . . 634
16.75.2.3 pet_penman() . . . 635
16.75.2.4 pet_priestly() . . . 637
16.75.2.5 sat_vap_pressure() . . . 638
16.75.2.6 slope_satpressure() . . . 639
16.76 mo_prepare_gridded_lai Module Reference . . . 640
16.76.1 Detailed Description . . . 640
16.76.2 Function/Subroutine Documentation . . . 641
16.76.2.1 prepare_gridded_daily_lai_data() . . . 641
16.76.2.2 prepare_gridded_mean_monthly_lai_data() . . . 642
16.77 mo_read_forcing_nc Module Reference . . . 643
16.77.1 Detailed Description . . . 643
16.77.2 Function/Subroutine Documentation . . . 644
16.77.2.1 get_time_vector_and_select() . . . 644
16.77.2.2 read_const_forcing_nc() . . . 645
16.77.2.3 read_forcing_nc() . . . 646
16.77.2.4 read_weights_nc() . . . 648
16.78 mo_read_latlon Module Reference . . . 649
16.78.1 Detailed Description . . . 649
16.78.2 Function/Subroutine Documentation . . . 650
16.78.2.1 read_latlon() . . . 650
16.79 mo_read_lut Module Reference . . . 651
16.79.1 Detailed Description . . . 651
16.79.2 Function/Subroutine Documentation . . . 652
16.79.2.1 read_geoformation_lut() . . . 652
16.79.2.2 read_lai_lut() . . . 653
16.80 mo_read_optional_data Module Reference . . . 654
16.80.1 Detailed Description . . . 654
16.80.2 Function/Subroutine Documentation . . . 654
16.80.2.1 read_basin_avg_tws() . . . 655
16.80.2.2 read_evapotranspiration() . . . 655
16.80.2.3 read_neutrons() . . . 656
16.80.2.4 read_soil_moisture() . . . 657
16.81 mo_read_spatial_data Module Reference . . . 658
16.81.1 Detailed Description . . . 659
16.81.2 Function/Subroutine Documentation . . . 659
16.81.2.1 read_header_ascii() . . . 659

16.81.2.2 read_spatial_data_ascii_dp() . . . 660
16.81.2.3 read_spatial_data_ascii_i4() . . . 661
16.82mo_read_timeseries Module Reference . . . 662
16.82.1 Detailed Description . . . 662
16.82.2 Function/Subroutine Documentation . . . 662
16.82.2.1 read_timeseries() . . . 663
16.83mo_read_wrapper Module Reference . . . 664
16.83.1 Detailed Description . . . 664
16.83.2 Function/Subroutine Documentation . . . 665
16.83.2.1 check_consistency_lut_map() . . . 665
16.83.2.2 read_data() . . . 666
16.84mo_restart Module Reference . . . 667
16.84.1 Detailed Description . . . 668
16.84.2 Function/Subroutine Documentation . . . 668
16.84.2.1 read_restart_states() . . . 668
16.84.2.2 unpack_field_and_write_1d_dp() . . . 669
16.84.2.3 unpack_field_and_write_1d_i4() . . . 669
16.84.2.4 unpack_field_and_write_2d_dp() . . . 670
16.84.2.5 unpack_field_and_write_3d_dp() . . . 670
16.84.2.6 write_restart_files() . . . 670
16.85mo_runoff Module Reference . . . 671
16.85.1 Detailed Description . . . 672
16.85.2 Function/Subroutine Documentation . . . 672
16.85.2.1 l1_total_runoff() . . . 672
16.85.2.2 runoff_sat_zone() . . . 673
16.85.2.3 runoff_unsat_zone() . . . 674
16.86mo_sce Module Reference . . . 675
16.86.1 Detailed Description . . . 675
16.86.2 Function/Subroutine Documentation . . . 675
16.86.2.1 cce() . . . 675
16.86.2.2 chkcst() . . . 676
16.86.2.3 comp() . . . 677
16.86.2.4 getpnt() . . . 677
16.86.2.5 parstt() . . . 678
16.86.2.6 sce() . . . 678
16.86.2.7 sort_matrix() . . . 682
16.87mo_set_netcdf_outputs Module Reference . . . 683
16.87.1 Detailed Description . . . 683
16.87.2 Function/Subroutine Documentation . . . 683
16.87.2.1 set_netcdf() . . . 683

16.88mo_snow_accum_melt Module Reference . . . 684
16.88.1 Detailed Description . . . 684
16.88.2 Function/Subroutine Documentation . . . 684
16.88.2.1 snow_accum_melt() . . . 684
16.89mo_soil_database Module Reference . . . 685
16.89.1 Detailed Description . . . 686
16.89.2 Function/Subroutine Documentation . . . 686
16.89.2.1 generate_soil_database() . . . 686
16.89.2.2 read_soil_lut() . . . 687
16.90mo_soil_moisture Module Reference . . . 688
16.90.1 Detailed Description . . . 688
16.90.2 Function/Subroutine Documentation . . . 688
16.90.2.1 feddes_et_reduction() . . . 689
16.90.2.2 jarvis_et_reduction() . . . 689
16.90.2.3 soil_moisture() . . . 690
16.91mo_spatial_agg_disagg_forcing Module Reference . . . 692
16.91.1 Detailed Description . . . 693
16.91.2 Function/Subroutine Documentation . . . 693
16.91.2.1 spatial_aggregation_3d() . . . 693
16.91.2.2 spatial_aggregation_4d() . . . 693
16.91.2.3 spatial_disaggregation_3d() . . . 694
16.91.2.4 spatial_disaggregation_4d() . . . 694
16.92mo_spatialsimilarity Module Reference . . . 694
16.92.1 Detailed Description . . . 694
16.92.2 Function/Subroutine Documentation . . . 695
16.92.2.1 nndv_dp() . . . 695
16.92.2.2 nndv_sp() . . . 695
16.92.2.3 pd_dp() . . . 695
16.92.2.4 pd_sp() . . . 695
16.93mo_standard_score Module Reference . . . 695
16.93.1 Detailed Description . . . 696
16.93.2 Function/Subroutine Documentation . . . 696
16.93.2.1 classified_standard_score_dp() . . . 696
16.93.2.2 classified_standard_score_sp() . . . 696
16.93.2.3 standard_score_dp() . . . 697
16.93.2.4 standard_score_sp() . . . 697
16.94mo_startup Module Reference . . . 697
16.94.1 Detailed Description . . . 697
16.94.2 Function/Subroutine Documentation . . . 697
16.94.2.1 constants_init() . . . 697

16.94.2.2 l2_variable_init() . . . 698
16.94.2.3 mhm_initialize() . . . 699
16.95 mo_string_utils Module Reference . . . 701
16.95.1 Detailed Description . . . 701
16.95.2 Function/Subroutine Documentation . . . 701
16.95.2.1 compress() . . . 702
16.95.2.2 divide_string() . . . 702
16.95.2.3 dp2str() . . . 703
16.95.2.4 equalstrings() . . . 703
16.95.2.5 i42str() . . . 703
16.95.2.6 i4array2str() . . . 704
16.95.2.7 i82str() . . . 704
16.95.2.8 log2str() . . . 704
16.95.2.9 nonull() . . . 704
16.95.2.10 sp2str() . . . 705
16.95.2.11 splitstring() . . . 705
16.95.2.12 startswith() . . . 705
16.95.2.13 str2num() . . . 706
16.95.2.14 tolower() . . . 706
16.95.2.15 toupper() . . . 707
16.95.3 Variable Documentation . . . 707
16.95.3.1 separator . . . 708
16.96 mo_template Module Reference . . . 708
16.96.1 Detailed Description . . . 708
16.96.2 Function/Subroutine Documentation . . . 708
16.96.2.1 circum() . . . 709
16.96.2.2 mean_dp() . . . 709
16.96.2.3 mean_sp() . . . 709
16.96.3 Variable Documentation . . . 709
16.96.3.1 itest . . . 709
16.96.3.2 pi_dp . . . 710
16.96.3.3 pi_sp . . . 710
16.97 mo_temporal_aggregation Module Reference . . . 710
16.97.1 Detailed Description . . . 710
16.97.2 Function/Subroutine Documentation . . . 710
16.97.2.1 day2mon_average_dp() . . . 711
16.97.2.2 hour2day_average_dp() . . . 711
16.98 mo_temporal_disagg_forcing Module Reference . . . 711
16.98.1 Detailed Description . . . 712
16.98.2 Function/Subroutine Documentation . . . 712

16.98.2.1 temporal_disagg_forcing() . . . 712
16.99 mo_timer Module Reference . . . 713
16.99.1 Detailed Description . . . 714
16.99.2 Function/Subroutine Documentation . . . 714
16.99.2.1 timer_check() . . . 714
16.99.2.2 timer_clear() . . . 715
16.99.2.3 timer_get() . . . 716
16.99.2.4 timer_print() . . . 717
16.99.2.5 timer_start() . . . 718
16.99.2.6 timer_stop() . . . 719
16.99.2.7 timers_init() . . . 720
16.99.3 Variable Documentation . . . 721
16.99.3.1 clock_rate . . . 721
16.99.3.2 cputime . . . 721
16.99.3.3 cycles1 . . . 721
16.99.3.4 cycles2 . . . 721
16.99.3.5 cycles_max . . . 722
16.99.3.6 max_timers . . . 722
16.99.3.7 status . . . 722
16.100 mo_upscaling_operators Module Reference . . . 722
16.100.1 Detailed Description . . . 723
16.100.2 Function/Subroutine Documentation . . . 723
16.100.2.1 l0_fractionalcover_in_lx() . . . 723
16.100.2.2 majority_statistics() . . . 724
16.100.2.3 upscale_arithmetic_mean() . . . 725
16.100.2.4 upscale_geometric_mean() . . . 726
16.100.2.5 upscale_harmonic_mean() . . . 727
16.100.2.6 upscale_p_norm() . . . 728
16.101 mo_utils Module Reference . . . 729
16.101.1 Detailed Description . . . 730
16.101.2 Function/Subroutine Documentation . . . 730
16.101.2.1 equal_dp() . . . 730
16.101.2.2 equal_sp() . . . 730
16.101.2.3 greaterequal_dp() . . . 730
16.101.2.4 greaterequal_sp() . . . 731
16.101.2.5 is_finite_dp() . . . 731
16.101.2.6 is_finite_sp() . . . 731
16.101.2.7 is_nan_dp() . . . 731
16.101.2.8 is_nan_sp() . . . 731
16.101.2.9 is_normal_dp() . . . 731

16.101.2.18 is_normal_sp() . . . 731
16.101.2.19 lesserequal_dp() . . . 731
16.101.2.20 lesserequal_sp() . . . 732
16.101.2.21 locate_0d_dp() . . . 732
16.101.2.22 locate_0d_sp() . . . 732
16.101.2.23 locate_1d_dp() . . . 732
16.101.2.24 locate_1d_sp() . . . 732
16.101.2.25 notequal_dp() . . . 732
16.101.2.26 notequal_sp() . . . 732
16.101.2.27 special_value_dp() . . . 733
16.101.2.28 special_value_sp() . . . 733
16.101.2.29 swap_vec_dp() . . . 733
16.101.2.30 swap_vec_i4() . . . 733
16.101.2.31 swap_vec_sp() . . . 734
16.101.2.32 swap_xy_dp() . . . 734
16.101.2.33 swap_xy_i4() . . . 734
16.101.2.34 swap_xy_sp() . . . 734
16.102 mo_write_ascii Module Reference . . . 734
16.102.1 Detailed Description . . . 734
16.102.2 Function/Subroutine Documentation . . . 735
16.102.2.1 write_configfile() . . . 735
16.102.2.2 write_optifile() . . . 736
16.102.2.3 write_optinamelist() . . . 737
16.103 mo_write_fluxes_states Module Reference . . . 738
16.103.1 Detailed Description . . . 739
16.103.2 Function/Subroutine Documentation . . . 739
16.103.2.1 close() . . . 739
16.103.2.2 createoutputfile() . . . 739
16.103.2.3 fluxesunit() . . . 740
16.103.2.4 newoutputdataset() . . . 741
16.103.2.5 newoutputvariable() . . . 742
16.103.2.6 updatedataset() . . . 743
16.103.2.7 updatevariable() . . . 745
16.103.2.8 writetimestep() . . . 745
16.103.2.9 writevariableattributes() . . . 746
16.103.2.10 writevariabletimestep() . . . 747
16.104 mo_xor4096 Module Reference . . . 747
16.104.1 Function/Subroutine Documentation . . . 748
16.104.1.1 get_timeseed_i4_0d() . . . 748
16.104.1.2 get_timeseed_i4_1d() . . . 748

16.104.1.3 get_timeseed_i8_0d() . . . 748
16.104.1.4 get_timeseed_i8_1d() . . . 748
16.104.1.5 xor4096d_0d() . . . 748
16.104.1.6 xor4096d_1d() . . . 748
16.104.1.7 xor4096f_0d() . . . 749
16.104.1.8 xor4096f_1d() . . . 749
16.104.1.9 xor4096gd_0d() . . . 749
16.104.1.10 xor4096gd_1d() . . . 749
16.104.1.11 xor4096gf_0d() . . . 749
16.104.1.12 xor4096gf_1d() . . . 750
16.104.1.13 xor4096l_0d() . . . 750
16.104.1.14 xor4096l_1d() . . . 750
16.104.1.15 xor4096s_0d() . . . 750
16.104.1.16 xor4096s_1d() . . . 750
16.104.2 Variable Documentation . . . 751
16.104.2.1 n_save_state . . . 751

17 Data Type Documentation **753**

17.1 mo_moment::absdev Interface Reference . . . 753
17.1.1 Member Function/Subroutine Documentation . . . 753
17.1.1.1 absdev_dp() . . . 753
17.1.1.2 absdev_sp() . . . 753
17.2 mo_anneal::anneal Interface Reference . . . 753
17.2.1 Detailed Description . . . 754
17.2.2 Member Function/Subroutine Documentation . . . 755
17.2.2.1 anneal_dp() . . . 755
17.3 mo_append::append Interface Reference . . . 756
17.3.1 Detailed Description . . . 756
17.3.2 Member Function/Subroutine Documentation . . . 758
17.3.2.1 append_char_3d() . . . 758
17.3.2.2 append_char_m_m() . . . 758
17.3.2.3 append_char_v_s() . . . 758
17.3.2.4 append_char_v_v() . . . 758
17.3.2.5 append_dp_3d() . . . 759
17.3.2.6 append_dp_m_m() . . . 759
17.3.2.7 append_dp_v_s() . . . 759
17.3.2.8 append_dp_v_v() . . . 759
17.3.2.9 append_i4_m_m() . . . 759
17.3.2.10 append_i4_v_s() . . . 759
17.3.2.11 append_i4_v_v() . . . 759

17.3.2.12 append_i8_3d() . . . 760
17.3.2.13 append_i8_m_m() . . . 760
17.3.2.14 append_i8_v_s() . . . 760
17.3.2.15 append_i8_v_v() . . . 760
17.3.2.16 append_lgt_3d() . . . 760
17.3.2.17 append_lgt_m_m() . . . 760
17.3.2.18 append_lgt_v_s() . . . 761
17.3.2.19 append_lgt_v_v() . . . 761
17.3.2.20 append_sp_3d() . . . 761
17.3.2.21 append_sp_m_m() . . . 761
17.3.2.22 append_sp_v_s() . . . 761
17.3.2.23 append_sp_v_v() . . . 761
17.4 mo_corr::arth Interface Reference . . . 761
17.4.1 Member Function/Subroutine Documentation . . . 762
17.4.1.1 arth_dp() . . . 762
17.4.1.2 arth_i4() . . . 762
17.4.1.3 arth_sp() . . . 762
17.5 mo_ncwrite::attribute Type Reference . . . 763
17.5.1 Member Data Documentation . . . 763
17.5.1.1 name . . . 763
17.5.1.2 nvalues . . . 763
17.5.1.3 values . . . 763
17.5.1.4 xtype . . . 763
17.6 mo_corr::autocoeffk Interface Reference . . . 764
17.6.1 Member Function/Subroutine Documentation . . . 764
17.6.1.1 autocoeffk_1d_dp() . . . 764
17.6.1.2 autocoeffk_1d_sp() . . . 764
17.6.1.3 autocoeffk_dp() . . . 764
17.6.1.4 autocoeffk_sp() . . . 764
17.7 mo_corr::autocorr Interface Reference . . . 765
17.7.1 Member Function/Subroutine Documentation . . . 765
17.7.1.1 autocorr_1d_dp() . . . 765
17.7.1.2 autocorr_1d_sp() . . . 765
17.7.1.3 autocorr_dp() . . . 765
17.7.1.4 autocorr_sp() . . . 765
17.8 mo_moment::average Interface Reference . . . 765
17.8.1 Member Function/Subroutine Documentation . . . 766
17.8.1.1 average_dp() . . . 766
17.8.1.2 average_sp() . . . 766
17.9 mo_mrm_global_variables::basininfo_mrm Type Reference . . . 766

17.9.1 Member Data Documentation . . . 767
17.9.1.1 gaugeidlist . . . 767
17.9.1.2 gaugeindexlist . . . 767
17.9.1.3 gaugenodelist . . . 767
17.9.1.4 inflowgaugeheadwater . . . 767
17.9.1.5 inflowgaugeidlist . . . 767
17.9.1.6 inflowgaugeindexlist . . . 767
17.9.1.7 inflowgaugenodelist . . . 768
17.9.1.8 l0_coloutlet . . . 768
17.9.1.9 l0_noutlet . . . 768
17.9.1.10 l0_rowoutlet . . . 768
17.9.1.11 ngauges . . . 768
17.9.1.12 ninflowgauges . . . 768
17.10 mo_errormeasures::bias Interface Reference . . . 768
17.10.1 Member Function/Subroutine Documentation . . . 768
17.10.1.1 bias_dp_1d() . . . 769
17.10.1.2 bias_dp_2d() . . . 769
17.10.1.3 bias_dp_3d() . . . 769
17.10.1.4 bias_sp_1d() . . . 769
17.10.1.5 bias_sp_2d() . . . 769
17.10.1.6 bias_sp_3d() . . . 769
17.11 mo_moment::central_moment Interface Reference . . . 770
17.11.1 Member Function/Subroutine Documentation . . . 770
17.11.1.1 central_moment_dp() . . . 770
17.11.1.2 central_moment_sp() . . . 770
17.12 mo_moment::central_moment_var Interface Reference . . . 770
17.12.1 Member Function/Subroutine Documentation . . . 770
17.12.1.1 central_moment_var_dp() . . . 770
17.12.1.2 central_moment_var_sp() . . . 771
17.13 mo_standard_score::classified_standard_score Interface Reference . . . 771
17.13.1 Detailed Description . . . 771
17.13.2 Member Function/Subroutine Documentation . . . 772
17.13.2.1 classified_standard_score_dp() . . . 772
17.13.2.2 classified_standard_score_sp() . . . 772
17.14 mo_corr::corr Interface Reference . . . 772
17.14.1 Member Function/Subroutine Documentation . . . 772
17.14.1.1 corr_dp() . . . 772
17.14.1.2 corr_sp() . . . 773
17.15 mo_moment::correlation Interface Reference . . . 773
17.15.1 Member Function/Subroutine Documentation . . . 773

17.15.1.1 correlation_dp() . . . 773
17.15.1.2 correlation_sp() . . . 773
17.16 mo_moment::covariance Interface Reference . . . 773
17.16.1 Member Function/Subroutine Documentation . . . 774
17.16.1.1 covariance_dp() . . . 774
17.16.1.2 covariance_sp() . . . 774
17.17 mo_corr::crosscoeffk Interface Reference . . . 774
17.17.1 Member Function/Subroutine Documentation . . . 774
17.17.1.1 crosscoeffk_dp() . . . 774
17.17.1.2 crosscoeffk_sp() . . . 775
17.18 mo_corr::crosscorr Interface Reference . . . 775
17.18.1 Member Function/Subroutine Documentation . . . 775
17.18.1.1 crosscorr_dp() . . . 775
17.18.1.2 crosscorr_sp() . . . 775
17.19 mo_orderpack::ctrper Interface Reference . . . 775
17.19.1 Member Function/Subroutine Documentation . . . 776
17.19.1.1 d_ctrper() . . . 776
17.19.1.2 i_ctrper() . . . 776
17.19.1.3 r_ctrper() . . . 776
17.20 mo_temporal_aggregation::day2mon_average Interface Reference . . . 776
17.20.1 Detailed Description . . . 776
17.20.2 Member Function/Subroutine Documentation . . . 777
17.20.2.1 day2mon_average_dp() . . . 777
17.21 mo_ncwrite::dims Type Reference . . . 777
17.21.1 Member Data Documentation . . . 778
17.21.1.1 dimid . . . 778
17.21.1.2 len . . . 778
17.21.1.3 name . . . 778
17.22 mo_ncwrite::dump_netcdf Interface Reference . . . 778
17.22.1 Member Function/Subroutine Documentation . . . 778
17.22.1.1 dump_netcdf_1d_dp() . . . 779
17.22.1.2 dump_netcdf_1d_i4() . . . 779
17.22.1.3 dump_netcdf_1d_sp() . . . 779
17.22.1.4 dump_netcdf_2d_dp() . . . 779
17.22.1.5 dump_netcdf_2d_i4() . . . 779
17.22.1.6 dump_netcdf_2d_sp() . . . 780
17.22.1.7 dump_netcdf_3d_dp() . . . 780
17.22.1.8 dump_netcdf_3d_i4() . . . 780
17.22.1.9 dump_netcdf_3d_sp() . . . 780
17.22.1.10 dump_netcdf_4d_dp() . . . 780

17.22.1.11 dump_netcdf_4d_i4() . . . 781
17.22.1.12 dump_netcdf_4d_sp() . . . 781
17.22.1.13 dump_netcdf_5d_dp() . . . 781
17.22.1.14 dump_netcdf_5d_i4() . . . 781
17.22.1.15 dump_netcdf_5d_sp() . . . 781
17.23 mo_utils::eq Interface Reference . . . 782
17.23.1 Member Function/Subroutine Documentation . . . 782
17.23.1.1 equal_dp() . . . 782
17.23.1.2 equal_sp() . . . 782
17.24 mo_utils::equal Interface Reference . . . 782
17.24.1 Detailed Description . . . 783
17.24.2 Member Function/Subroutine Documentation . . . 783
17.24.2.1 equal_dp() . . . 783
17.24.2.2 equal_sp() . . . 783
17.25 mo_optimization_utils::eval_interface Interface Reference . . . 783
17.25.1 Constructor & Destructor Documentation . . . 784
17.25.1.1 eval_interface() . . . 784
17.26 mo_orderpack::fndnth Interface Reference . . . 784
17.26.1 Member Function/Subroutine Documentation . . . 784
17.26.1.1 d_fndnth() . . . 784
17.26.1.2 i_fndnth() . . . 784
17.26.1.3 r_fndnth() . . . 784
17.27 mo_corr::four1 Interface Reference . . . 785
17.27.1 Member Function/Subroutine Documentation . . . 785
17.27.1.1 four1_dp() . . . 785
17.27.1.2 four1_sp() . . . 785
17.28 mo_corr::fourrow Interface Reference . . . 785
17.28.1 Member Function/Subroutine Documentation . . . 785
17.28.1.1 fourrow_dp() . . . 785
17.28.1.2 fourrow_sp() . . . 786
17.29 mo_mrm_global_variables::gaugingstation Type Reference . . . 786
17.29.1 Member Data Documentation . . . 786
17.29.1.1 basinid . . . 786
17.29.1.2 fname . . . 786
17.29.1.3 gaugeid . . . 787
17.29.1.4 q . . . 787
17.30 mo_utils::ge Interface Reference . . . 787
17.30.1 Member Function/Subroutine Documentation . . . 787
17.30.1.1 greaterequal_dp() . . . 787
17.30.1.2 greaterequal_sp() . . . 787

17.31 mo_anneal::generate_neighborhood_weight Interface Reference . . . 787
17.31.1 Member Function/Subroutine Documentation . . . 787
17.31.1.1 generate_neighborhood_weight_dp() . . . 788
17.32 mo_ncread::get_ncvar Interface Reference . . . 788
17.32.1 Member Function/Subroutine Documentation . . . 788
17.32.1.1 get_ncvar_0d_dp() . . . 788
17.32.1.2 get_ncvar_0d_i1() . . . 789
17.32.1.3 get_ncvar_0d_i4() . . . 789
17.32.1.4 get_ncvar_0d_sp() . . . 789
17.32.1.5 get_ncvar_1d_dp() . . . 789
17.32.1.6 get_ncvar_1d_i1() . . . 789
17.32.1.7 get_ncvar_1d_i4() . . . 789
17.32.1.8 get_ncvar_1d_sp() . . . 790
17.32.1.9 get_ncvar_2d_dp() . . . 790
17.32.1.10 get_ncvar_2d_i1() . . . 790
17.32.1.11 get_ncvar_2d_i4() . . . 790
17.32.1.12 get_ncvar_2d_sp() . . . 791
17.32.1.13 get_ncvar_3d_dp() . . . 791
17.32.1.14 get_ncvar_3d_i1() . . . 791
17.32.1.15 get_ncvar_3d_i4() . . . 791
17.32.1.16 get_ncvar_3d_sp() . . . 791
17.32.1.17 get_ncvar_4d_dp() . . . 792
17.32.1.18 get_ncvar_4d_i1() . . . 792
17.32.1.19 get_ncvar_4d_i4() . . . 792
17.32.1.20 get_ncvar_4d_sp() . . . 792
17.32.1.21 get_ncvar_5d_dp() . . . 792
17.32.1.22 get_ncvar_5d_i1() . . . 793
17.32.1.23 get_ncvar_5d_i4() . . . 793
17.32.1.24 get_ncvar_5d_sp() . . . 793
17.33 mo_xor4096::get_timeseed Interface Reference . . . 793
17.33.1 Member Function/Subroutine Documentation . . . 793
17.33.1.1 get_timeseed_i4_0d() . . . 794
17.33.1.2 get_timeseed_i4_1d() . . . 794
17.33.1.3 get_timeseed_i8_0d() . . . 794
17.33.1.4 get_timeseed_i8_1d() . . . 794
17.34 mo_anneal::gettemperature Interface Reference . . . 794
17.34.1 Detailed Description . . . 794
17.34.2 Member Function/Subroutine Documentation . . . 795
17.34.2.1 gettemperature_dp() . . . 795
17.35 mo_utils::greaterequal Interface Reference . . . 796

17.35.1 Member Function/Subroutine Documentation . . . 796
17.35.1.1 greaterequal_dp() . . . 796
17.35.1.2 greaterequal_sp() . . . 796
17.36 mo_common_variables::grid Type Reference . . . 797
17.36.1 Member Data Documentation . . . 797
17.36.1.1 cellarea . . . 797
17.36.1.2 cellcoor . . . 798
17.36.1.3 cellsize . . . 798
17.36.1.4 id . . . 798
17.36.1.5 iend . . . 798
17.36.1.6 istart . . . 798
17.36.1.7 mask . . . 798
17.36.1.8 ncells . . . 798
17.36.1.9 ncols . . . 798
17.36.1.10 nodata_value . . . 798
17.36.1.11 nrows . . . 799
17.36.1.12 x . . . 799
17.36.1.13 xllcorner . . . 799
17.36.1.14 y . . . 799
17.36.1.15 yllcorner . . . 799
17.37 mo_common_variables::gridremapper Type Reference . . . 800
17.37.1 Member Data Documentation . . . 800
17.37.1.1 high_res_grid . . . 800
17.37.1.2 left_bound . . . 801
17.37.1.3 low_res_grid . . . 801
17.37.1.4 lower_bound . . . 801
17.37.1.5 lowres_id_on_highres . . . 801
17.37.1.6 n_subcells . . . 801
17.37.1.7 right_bound . . . 801
17.37.1.8 upper_bound . . . 801
17.38 mo_temporal_aggregation::hour2day_average Interface Reference . . . 801
17.38.1 Detailed Description . . . 802
17.38.2 Member Function/Subroutine Documentation . . . 802
17.38.2.1 hour2day_average_dp() . . . 802
17.39 mo_orderpack::indmed Interface Reference . . . 802
17.39.1 Member Function/Subroutine Documentation . . . 803
17.39.1.1 d_indmed() . . . 803
17.39.1.2 i_indmed() . . . 803
17.39.1.3 r_indmed() . . . 803
17.40 mo_orderpack::indnth Interface Reference . . . 803

17.40.1 Member Function/Subroutine Documentation . . . 803
17.40.1.1 d_indnth() . . . 803
17.40.1.2 i_indnth() . . . 804
17.40.1.3 r_indnth() . . . 804
17.41 mo_orderpack::inspar Interface Reference . . . 804
17.41.1 Member Function/Subroutine Documentation . . . 804
17.41.1.1 d_inspar() . . . 804
17.41.1.2 i_inspar() . . . 804
17.41.1.3 r_inspar() . . . 804
17.42 mo_orderpack::inssor Interface Reference . . . 805
17.42.1 Member Function/Subroutine Documentation . . . 805
17.42.1.1 d_inssor() . . . 805
17.42.1.2 i_inssor() . . . 805
17.42.1.3 r_inssor() . . . 805
17.43 mo_utils::is_finite Interface Reference . . . 805
17.43.1 Detailed Description . . . 805
17.43.2 Member Function/Subroutine Documentation . . . 806
17.43.2.1 is_finite_dp() . . . 806
17.43.2.2 is_finite_sp() . . . 806
17.44 mo_utils::is_nan Interface Reference . . . 806
17.44.1 Member Function/Subroutine Documentation . . . 806
17.44.1.1 is_nan_dp() . . . 806
17.44.1.2 is_nan_sp() . . . 807
17.45 mo_utils::is_normal Interface Reference . . . 807
17.45.1 Member Function/Subroutine Documentation . . . 807
17.45.1.1 is_normal_dp() . . . 807
17.45.1.2 is_normal_sp() . . . 807
17.46 mo_errormeasures::kge Interface Reference . . . 807
17.46.1 Detailed Description . . . 808
17.46.2 Member Function/Subroutine Documentation . . . 808
17.46.2.1 kge_dp_1d() . . . 808
17.46.2.2 kge_dp_2d() . . . 809
17.46.2.3 kge_dp_3d() . . . 809
17.46.2.4 kge_sp_1d() . . . 809
17.46.2.5 kge_sp_2d() . . . 809
17.46.2.6 kge_sp_3d() . . . 809
17.47 mo_errormeasures::kgenocorr Interface Reference . . . 809
17.47.1 Detailed Description . . . 810
17.47.2 Member Function/Subroutine Documentation . . . 810
17.47.2.1 kgenocorr_dp_1d() . . . 810

17.47.2.2 kgenocorr_dp_2d() . . . 811
17.47.2.3 kgenocorr_dp_3d() . . . 811
17.47.2.4 kgenocorr_sp_1d() . . . 811
17.47.2.5 kgenocorr_sp_2d() . . . 811
17.47.2.6 kgenocorr_sp_3d() . . . 811
17.48 mo_moment::kurtosis Interface Reference . . . 811
17.48.1 Member Function/Subroutine Documentation . . . 812
17.48.1.1 kurtosis_dp() . . . 812
17.48.1.2 kurtosis_sp() . . . 812
17.49 mo_utils::le Interface Reference . . . 812
17.49.1 Member Function/Subroutine Documentation . . . 812
17.49.1.1 lesserequal_dp() . . . 812
17.49.1.2 lesserequal_sp() . . . 812
17.50 mo_utils::lesserequal Interface Reference . . . 813
17.50.1 Member Function/Subroutine Documentation . . . 813
17.50.1.1 lesserequal_dp() . . . 813
17.50.1.2 lesserequal_sp() . . . 813
17.51 mo_linfit::linfit Interface Reference . . . 813
17.51.1 Detailed Description . . . 813
17.51.2 Member Function/Subroutine Documentation . . . 814
17.51.2.1 linfit_dp() . . . 814
17.51.2.2 linfit_sp() . . . 814
17.52 mo_errormeasures::lnnse Interface Reference . . . 814
17.52.1 Member Function/Subroutine Documentation . . . 815
17.52.1.1 lnnse_dp_1d() . . . 815
17.52.1.2 lnnse_dp_2d() . . . 815
17.52.1.3 lnnse_dp_3d() . . . 815
17.52.1.4 lnnse_sp_1d() . . . 815
17.52.1.5 lnnse_sp_2d() . . . 815
17.52.1.6 lnnse_sp_3d() . . . 815
17.53 mo_utils::locate Interface Reference . . . 816
17.53.1 Detailed Description . . . 816
17.53.2 Member Function/Subroutine Documentation . . . 816
17.53.2.1 locate_0d_dp() . . . 816
17.53.2.2 locate_0d_sp() . . . 817
17.53.2.3 locate_1d_dp() . . . 817
17.53.2.4 locate_1d_sp() . . . 817
17.54 mo_mad::mad Interface Reference . . . 817
17.54.1 Member Function/Subroutine Documentation . . . 817
17.54.1.1 mad_dp() . . . 817

17.54.1.2 mad_sp() . . . 817
17.54.1.3 mad_val_dp() . . . 818
17.54.1.4 mad_val_sp() . . . 818
17.55 mo_errormeasures::mae Interface Reference . . . 818
17.55.1 Member Function/Subroutine Documentation . . . 818
17.55.1.1 mae_dp_1d() . . . 818
17.55.1.2 mae_dp_2d() . . . 819
17.55.1.3 mae_dp_3d() . . . 819
17.55.1.4 mae_sp_1d() . . . 819
17.55.1.5 mae_sp_2d() . . . 819
17.55.1.6 mae_sp_3d() . . . 819
17.56 mo_mcmc::mcmc Interface Reference . . . 819
17.56.1 Detailed Description . . . 820
17.56.2 Member Function/Subroutine Documentation . . . 823
17.56.2.1 mcmc_dp() . . . 823
17.57 mo_mcmc::mcmc_stddev Interface Reference . . . 824
17.57.1 Detailed Description . . . 824
17.57.2 Member Function/Subroutine Documentation . . . 827
17.57.2.1 mcmc_stddev_dp() . . . 827
17.58 mo_moment::mean Interface Reference . . . 828
17.58.1 Member Function/Subroutine Documentation . . . 828
17.58.1.1 mean_dp() . . . 828
17.58.1.2 mean_sp() . . . 828
17.59 mo_template::mean Interface Reference . . . 828
17.59.1 Detailed Description . . . 828
17.59.2 Member Function/Subroutine Documentation . . . 829
17.59.2.1 mean_dp() . . . 829
17.59.2.2 mean_sp() . . . 829
17.60 mo_percentile::median Interface Reference . . . 829
17.60.1 Member Function/Subroutine Documentation . . . 829
17.60.1.1 median_dp() . . . 830
17.60.1.2 median_sp() . . . 830
17.61 mo_moment::mixed_central_moment Interface Reference . . . 830
17.61.1 Member Function/Subroutine Documentation . . . 830
17.61.1.1 mixed_central_moment_dp() . . . 830
17.61.1.2 mixed_central_moment_sp() . . . 830
17.62 mo_moment::mixed_central_moment_var Interface Reference . . . 831
17.62.1 Member Function/Subroutine Documentation . . . 831
17.62.1.1 mixed_central_moment_var_dp() . . . 831
17.62.1.2 mixed_central_moment_var_sp() . . . 831

17.63mo_moment::moment Interface Reference . . . 831
17.63.1 Member Function/Subroutine Documentation . . . 831
17.63.1.1 moment_dp() . . . 831
17.63.1.2 moment_sp() . . . 832
17.64mo_orderpack::mrgref Interface Reference . . . 832
17.64.1 Member Function/Subroutine Documentation . . . 832
17.64.1.1 d_mrgref() . . . 832
17.64.1.2 i_mrgref() . . . 832
17.64.1.3 r_mrgref() . . . 833
17.65mo_orderpack::mrgrnk Interface Reference . . . 833
17.65.1 Member Function/Subroutine Documentation . . . 833
17.65.1.1 d_mrgrnk() . . . 833
17.65.1.2 i_mrgrnk() . . . 833
17.65.1.3 r_mrgrnk() . . . 833
17.66mo_errormeasures::mse Interface Reference . . . 833
17.66.1 Member Function/Subroutine Documentation . . . 834
17.66.1.1 mse_dp_1d() . . . 834
17.66.1.2 mse_dp_2d() . . . 834
17.66.1.3 mse_dp_3d() . . . 834
17.66.1.4 mse_sp_1d() . . . 834
17.66.1.5 mse_sp_2d() . . . 834
17.66.1.6 mse_sp_3d() . . . 835
17.67mo_orderpack::mulcnt Interface Reference . . . 835
17.67.1 Member Function/Subroutine Documentation . . . 835
17.67.1.1 d_mulcnt() . . . 835
17.67.1.2 i_mulcnt() . . . 835
17.67.1.3 r_mulcnt() . . . 835
17.68mo_percentile::n_element Interface Reference . . . 835
17.68.1 Member Function/Subroutine Documentation . . . 836
17.68.1.1 n_element_dp() . . . 836
17.68.1.2 n_element_sp() . . . 836
17.69mo_netcdf::ncdataset Interface Reference . . . 836
17.69.1 Detailed Description . . . 838
17.69.2 Member Function/Subroutine Documentation . . . 839
17.69.2.1 close() . . . 839
17.69.2.2 getattribute() . . . 839
17.69.2.3 getdimension() . . . 839
17.69.2.4 getdimensionbyid() . . . 840
17.69.2.5 getdimensionbyname() . . . 840
17.69.2.6 getglobalattributechar() . . . 840

17.69.2.7 getglobalattributef32() . . . 840
17.69.2.8 getglobalattributef64() . . . 840
17.69.2.9 getglobalattributei16() . . . 840
17.69.2.10 getglobalattributei32() . . . 840
17.69.2.11 getglobalattributei64() . . . 841
17.69.2.12 getglobalattributei8() . . . 841
17.69.2.13 getnovariables() . . . 841
17.69.2.14 getunlimiteddimension() . . . 841
17.69.2.15 getvariable() . . . 841
17.69.2.16 getvariablebyname() . . . 842
17.69.2.17 getvariableids() . . . 842
17.69.2.18 getvariables() . . . 842
17.69.2.19 hasdimension() . . . 842
17.69.2.20 hasvariable() . . . 843
17.69.2.21 initncdataset() . . . 843
17.69.2.22 isunlimited() . . . 843
17.69.2.23 setattribute() . . . 843
17.69.2.24 setdimension() . . . 844
17.69.2.25 setglobalattributechar() . . . 844
17.69.2.26 setglobalattributef32() . . . 844
17.69.2.27 setglobalattributef64() . . . 845
17.69.2.28 setglobalattributei16() . . . 845
17.69.2.29 setglobalattributei32() . . . 845
17.69.2.30 setglobalattributei64() . . . 845
17.69.2.31 setglobalattributei8() . . . 845
17.69.2.32 setvariable() . . . 845
17.69.2.33 setvariablewithids() . . . 846
17.69.2.34 setvariablewithnames() . . . 846
17.69.2.35 setvariablewithtypes() . . . 846
17.69.3 Member Data Documentation . . . 846
17.69.3.1 dataset . . . 846
17.69.3.2 file . . . 846
17.69.3.3 filename . . . 846
17.69.3.4 fname . . . 846
17.69.3.5 id . . . 847
17.69.3.6 mode . . . 847
17.69.3.7 netcdf . . . 847
17.69.3.8 of . . . 847
17.69.3.9 open . . . 847
17.69.3.10 opened . . . 847

17.69.3.11 the . . . 847
17.70 mo_netcdf::ncdimension Type Reference . . . 847
17.70.1 Detailed Description . . . 851
17.70.2 Member Function/Subroutine Documentation . . . 851
17.70.2.1 getattribute() . . . 851
17.70.2.2 getdata() . . . 852
17.70.2.3 getdata1df32() . . . 852
17.70.2.4 getdata1df64() . . . 852
17.70.2.5 getdata1di16() . . . 852
17.70.2.6 getdata1di32() . . . 852
17.70.2.7 getdata1di64() . . . 853
17.70.2.8 getdata1di8() . . . 853
17.70.2.9 getdata2df32() . . . 853
17.70.2.10 getdata2df64() . . . 853
17.70.2.11 getdata2di16() . . . 853
17.70.2.12 getdata2di32() . . . 853
17.70.2.13 getdata2di64() . . . 853
17.70.2.14 getdata2di8() . . . 853
17.70.2.15 getdata3df32() . . . 853
17.70.2.16 getdata3df64() . . . 854
17.70.2.17 getdata3di16() . . . 854
17.70.2.18 getdata3di32() . . . 854
17.70.2.19 getdata3di64() . . . 854
17.70.2.20 getdata3di8() . . . 854
17.70.2.21 getdata4df32() . . . 854
17.70.2.22 getdata4df64() . . . 854
17.70.2.23 getdata4di16() . . . 854
17.70.2.24 getdata4di32() . . . 854
17.70.2.25 getdata4di64() . . . 855
17.70.2.26 getdata4di8() . . . 855
17.70.2.27 getdata5df32() . . . 855
17.70.2.28 getdata5df64() . . . 855
17.70.2.29 getdata5di16() . . . 855
17.70.2.30 getdata5di32() . . . 855
17.70.2.31 getdata5di64() . . . 855
17.70.2.32 getdata5di8() . . . 855
17.70.2.33 getdatascalarf32() . . . 855
17.70.2.34 getdatascalarf64() . . . 856
17.70.2.35 getdatascalari16() . . . 856
17.70.2.36 getdatascalari32() . . . 856

17.70.2.37 getdatascalari64() . . . 856
17.70.2.38 getdatascalari8() . . . 856
17.70.2.39 getdimensions() . . . 856
17.70.2.40 getdtype() . . . 856
17.70.2.41 getfillvalue() . . . 856
17.70.2.42 getname() . . . 857
17.70.2.43 getnodimensions() . . . 857
17.70.2.44 getshape() . . . 857
17.70.2.45 getvariableattributechar() . . . 857
17.70.2.46 getvariableattributef32() . . . 857
17.70.2.47 getvariableattributef64() . . . 857
17.70.2.48 getvariableattributei16() . . . 857
17.70.2.49 getvariableattributei32() . . . 858
17.70.2.50 getvariableattributei64() . . . 858
17.70.2.51 getvariableattributei8() . . . 858
17.70.2.52 getvariablefillvaluef32() . . . 858
17.70.2.53 getvariablefillvaluef64() . . . 858
17.70.2.54 getvariablefillvaluei16() . . . 858
17.70.2.55 getvariablefillvaluei32() . . . 858
17.70.2.56 getvariablefillvaluei64() . . . 858
17.70.2.57 getvariablefillvaluei8() . . . 858
17.70.2.58 hasattribute() . . . 859
17.70.2.59 initncvariable() . . . 859
17.70.2.60 isunlimited() . . . 859
17.70.2.61 setattribute() . . . 859
17.70.2.62 setdata() . . . 860
17.70.2.63 setdata1df32() . . . 860
17.70.2.64 setdata1df64() . . . 860
17.70.2.65 setdata1di16() . . . 860
17.70.2.66 setdata1di32() . . . 860
17.70.2.67 setdata1di64() . . . 860
17.70.2.68 setdata1di8() . . . 861
17.70.2.69 setdata2df32() . . . 861
17.70.2.70 setdata2df64() . . . 861
17.70.2.71 setdata2di16() . . . 861
17.70.2.72 setdata2di32() . . . 861
17.70.2.73 setdata2di64() . . . 861
17.70.2.74 setdata2di8() . . . 861
17.70.2.75 setdata3df32() . . . 861
17.70.2.76 setdata3df64() . . . 861

17.70.2.77 setdata3di16() . . . 862
17.70.2.78 setdata3di32() . . . 862
17.70.2.79 setdata3di64() . . . 862
17.70.2.80 setdata3di8() . . . 862
17.70.2.81 setdata4df32() . . . 862
17.70.2.82 setdata4df64() . . . 862
17.70.2.83 setdata4di16() . . . 862
17.70.2.84 setdata4di32() . . . 862
17.70.2.85 setdata4di64() . . . 862
17.70.2.86 setdata4di8() . . . 863
17.70.2.87 setdata5df32() . . . 863
17.70.2.88 setdata5df64() . . . 863
17.70.2.89 setdata5di16() . . . 863
17.70.2.90 setdata5di32() . . . 863
17.70.2.91 setdata5di64() . . . 863
17.70.2.92 setdata5di8() . . . 863
17.70.2.93 setdatascalarf32() . . . 863
17.70.2.94 setdatascalarf64() . . . 863
17.70.2.95 setdatascalari16() . . . 864
17.70.2.96 setdatascalari32() . . . 864
17.70.2.97 setdatascalari64() . . . 864
17.70.2.98 setdatascalari8() . . . 864
17.70.2.99 setfillvalue() . . . 864
17.70.2.100 setvariableattributechar() . . . 864
17.70.2.101 setvariableattributef32() . . . 865
17.70.2.102 setvariableattributef64() . . . 865
17.70.2.103 setvariableattributei16() . . . 865
17.70.2.104 setvariableattributei32() . . . 865
17.70.2.105 setvariableattributei64() . . . 865
17.70.2.106 setvariableattributei8() . . . 865
17.70.2.107 setvariablefillvaluef32() . . . 865
17.70.2.108 setvariablefillvaluef64() . . . 865
17.70.2.109 setvariablefillvaluei16() . . . 865
17.70.2.110 setvariablefillvaluei32() . . . 866
17.70.2.111 setvariablefillvaluei64() . . . 866
17.70.2.112 setvariablefillvaluei8() . . . 866
17.70.3 Member Data Documentation . . . 866
17.70.3.1 dimension [1/2] . . . 866
17.70.3.2 dimension [2/2] . . . 866
17.70.3.3 id . . . 866

17.70.3.4 netcdf . . . 866
17.70.3.5 parent . . . 866
17.70.3.6 s . . . 866
17.70.3.7 the [1/2] . . . 867
17.70.3.8 the [2/2] . . . 867
17.71 mo_netcdf::ncvariable Interface Reference . . . 867
17.72 mo_utils::ne Interface Reference . . . 867
17.72.1 Member Function/Subroutine Documentation . . . 867
17.72.1.1 notequal_dp() . . . 867
17.72.1.2 notequal_sp() . . . 867
17.73 mo_orderpack::nearless Interface Reference . . . 868
17.73.1 Member Function/Subroutine Documentation . . . 868
17.73.1.1 d_nearless() . . . 868
17.73.1.2 i_nearless() . . . 868
17.73.1.3 r_nearless() . . . 868
17.74 mo_spatialsimilarity::nndv Interface Reference . . . 868
17.74.1 Detailed Description . . . 868
17.74.2 Member Function/Subroutine Documentation . . . 870
17.74.2.1 nndv_dp() . . . 870
17.74.2.2 nndv_sp() . . . 870
17.75 mo_utils::notequal Interface Reference . . . 870
17.75.1 Member Function/Subroutine Documentation . . . 870
17.75.1.1 notequal_dp() . . . 870
17.75.1.2 notequal_sp() . . . 871
17.76 mo_errormeasures::nse Interface Reference . . . 871
17.76.1 Member Function/Subroutine Documentation . . . 871
17.76.1.1 nse_dp_1d() . . . 871
17.76.1.2 nse_dp_2d() . . . 871
17.76.1.3 nse_dp_3d() . . . 871
17.76.1.4 nse_sp_1d() . . . 872
17.76.1.5 nse_sp_2d() . . . 872
17.76.1.6 nse_sp_3d() . . . 872
17.77 mo_string_utils::num2str Interface Reference . . . 872
17.77.1 Detailed Description . . . 872
17.77.2 Member Function/Subroutine Documentation . . . 873
17.77.2.1 dp2str() . . . 873
17.77.2.2 i42str() . . . 873
17.77.2.3 i82str() . . . 873
17.77.2.4 log2str() . . . 873
17.77.2.5 sp2str() . . . 874

17.78 mo_string_utils::numarray2str Interface Reference . . . 874
17.78.1 Detailed Description . . . 874
17.78.2 Member Function/Subroutine Documentation . . . 874
17.78.2.1 i4array2str() . . . 874
17.79 mo_optimization_utils::objective_interface Interface Reference . . . 875
17.79.1 Constructor & Destructor Documentation . . . 875
17.79.1.1 objective_interface() . . . 875
17.80 mo_orderpack::omedian Interface Reference . . . 875
17.80.1 Member Function/Subroutine Documentation . . . 875
17.80.1.1 d_median() . . . 875
17.80.1.2 i_median() . . . 875
17.80.1.3 r_median() . . . 876
17.81 mo_mrm_write_fluxes_states::outputdataset Interface Reference . . . 876
17.81.1 Member Function/Subroutine Documentation . . . 877
17.81.1.1 close() . . . 877
17.81.1.2 updatedataset() . . . 877
17.81.1.3 writetimestep() . . . 877
17.81.2 Member Data Documentation . . . 877
17.81.2.1 all . . . 877
17.81.2.2 basin . . . 877
17.81.2.3 count . . . 878
17.81.2.4 counter . . . 878
17.81.2.5 created . . . 878
17.81.2.6 ibasin . . . 878
17.81.2.7 id . . . 878
17.81.2.8 nc . . . 878
17.81.2.9 ncdataset . . . 878
17.81.2.10 steps . . . 878
17.81.2.11 store . . . 878
17.81.2.12 time . . . 879
17.81.2.13 to . . . 879
17.81.2.14 variables . . . 879
17.81.2.15 vars . . . 879
17.81.2.16 write . . . 879
17.81.2.17 written . . . 879
17.82 mo_write_fluxes_states::outputdataset Interface Reference . . . 880
17.82.1 Member Function/Subroutine Documentation . . . 881
17.82.1.1 close() . . . 881
17.82.1.2 updatedataset() . . . 881
17.82.1.3 writetimestep() . . . 881

17.82.2 Member Data Documentation . . . 881
17.82.2.1 all . . . 881
17.82.2.2 basin . . . 881
17.82.2.3 count . . . 882
17.82.2.4 counter . . . 882
17.82.2.5 created . . . 882
17.82.2.6 ibasin . . . 882
17.82.2.7 id . . . 882
17.82.2.8 nc . . . 882
17.82.2.9 ncdataset . . . 882
17.82.2.10 steps . . . 882
17.82.2.11 store . . . 882
17.82.2.12 time . . . 883
17.82.2.13 to . . . 883
17.82.2.14 variables . . . 883
17.82.2.15 vars . . . 883
17.82.2.16 write . . . 883
17.82.2.17 written . . . 883
17.83 mo_mrm_write_fluxes_states::outputvariable Interface Reference . . . 884
17.83.1 Member Function/Subroutine Documentation . . . 885
17.83.1.1 updatevariable() . . . 885
17.83.1.2 writevariabletimestep() . . . 885
17.83.2 Member Data Documentation . . . 885
17.83.2.1 average . . . 885
17.83.2.2 avg . . . 885
17.83.2.3 before . . . 886
17.83.2.4 between . . . 886
17.83.2.5 calls . . . 886
17.83.2.6 contains . . . 886
17.83.2.7 count . . . 886
17.83.2.8 counter . . . 886
17.83.2.9 data [1/2] . . . 886
17.83.2.10 data [2/2] . . . 886
17.83.2.11 mask . . . 886
17.83.2.12 nc . . . 887
17.83.2.13 ncdataset . . . 887
17.83.2.14 number . . . 887
17.83.2.15 of . . . 887
17.83.2.16 reconstruct . . . 887
17.83.2.17 store . . . 887

17.83.2.18 the [1/3] . . . 887
17.83.2.19 the [2/3] . . . 887
17.83.2.20 the [3/3] . . . 887
17.83.2.21 to . . . 888
17.83.2.22 updatevariable . . . 888
17.83.2.23 variable . . . 888
17.83.2.24 which . . . 888
17.83.2.25 writes . . . 888
17.83.2.26 writing . . . 888
17.84 mo_write_fluxes_states::outputvariable Interface Reference . . . 889
17.84.1 Member Function/Subroutine Documentation . . . 890
17.84.1.1 updatevariable() . . . 890
17.84.1.2 writevariabletimestep() . . . 890
17.84.2 Member Data Documentation . . . 890
17.84.2.1 average . . . 890
17.84.2.2 avg . . . 890
17.84.2.3 before . . . 891
17.84.2.4 between . . . 891
17.84.2.5 calls . . . 891
17.84.2.6 contains . . . 891
17.84.2.7 count . . . 891
17.84.2.8 counter . . . 891
17.84.2.9 data [1/2] . . . 891
17.84.2.10 data [2/2] . . . 891
17.84.2.11 mask . . . 891
17.84.2.12 nc . . . 892
17.84.2.13 ncdataset . . . 892
17.84.2.14 number . . . 892
17.84.2.15 of . . . 892
17.84.2.16 reconstruct . . . 892
17.84.2.17 store . . . 892
17.84.2.18 the [1/3] . . . 892
17.84.2.19 the [2/3] . . . 892
17.84.2.20 the [3/3] . . . 892
17.84.2.21 to . . . 893
17.84.2.22 updatevariable . . . 893
17.84.2.23 variable . . . 893
17.84.2.24 which . . . 893
17.84.2.25 writes . . . 893
17.84.2.26 writing . . . 893

17.85mo_append::paste Interface Reference . . . 893
17.85.1 Detailed Description . . . 894
17.85.2 Member Function/Subroutine Documentation . . . 894
17.85.2.1 paste_char_m_m() . . . 894
17.85.2.2 paste_char_m_s() . . . 894
17.85.2.3 paste_char_m_v() . . . 895
17.85.2.4 paste_dp_m_m() . . . 895
17.85.2.5 paste_dp_m_s() . . . 895
17.85.2.6 paste_dp_m_v() . . . 895
17.85.2.7 paste_i4_m_m() . . . 895
17.85.2.8 paste_i4_m_s() . . . 895
17.85.2.9 paste_i4_m_v() . . . 896
17.85.2.10paste_i8_m_m() . . . 896
17.85.2.11paste_i8_m_s() . . . 896
17.85.2.12paste_i8_m_v() . . . 896
17.85.2.13paste_lgt_m_m() . . . 896
17.85.2.14paste_lgt_m_s() . . . 896
17.85.2.15paste_lgt_m_v() . . . 896
17.85.2.16paste_sp_m_m() . . . 897
17.85.2.17paste_sp_m_s() . . . 897
17.85.2.18paste_sp_m_v() . . . 897
17.86mo_spatialsimilarity::pd Interface Reference . . . 897
17.86.1 Detailed Description . . . 897
17.86.2 Member Function/Subroutine Documentation . . . 898
17.86.2.1 pd_dp() . . . 899
17.86.2.2 pd_sp() . . . 899
17.87mo_percentile::percentile Interface Reference . . . 899
17.87.1 Member Function/Subroutine Documentation . . . 899
17.87.1.1 percentile_0d_dp() . . . 899
17.87.1.2 percentile_0d_sp() . . . 899
17.87.1.3 percentile_1d_dp() . . . 900
17.87.1.4 percentile_1d_sp() . . . 900
17.88mo_common_variables::period Type Reference . . . 900
17.88.1 Member Data Documentation . . . 901
17.88.1.1 dend . . . 901
17.88.1.2 dstart . . . 901
17.88.1.3 julend . . . 901
17.88.1.4 julstart . . . 901
17.88.1.5 mend . . . 901
17.88.1.6 mstart . . . 901

17.88.1.7 nobs . . . 901
17.88.1.8 yend . . . 902
17.88.1.9 ystart . . . 902
17.89 mo_percentile::qmedian Interface Reference . . . 902
17.89.1 Member Function/Subroutine Documentation . . . 902
17.89.1.1 qmedian_dp() . . . 902
17.89.1.2 qmedian_sp() . . . 902
17.90 mo_orderpack::rapknr Interface Reference . . . 902
17.90.1 Member Function/Subroutine Documentation . . . 902
17.90.1.1 d_rapknr() . . . 903
17.90.1.2 i_rapknr() . . . 903
17.90.1.3 r_rapknr() . . . 903
17.91 mo_read_spatial_data::read_spatial_data_ascii Interface Reference . . . 903
17.91.1 Detailed Description . . . 903
17.91.2 Member Function/Subroutine Documentation . . . 904
17.91.2.1 read_spatial_data_ascii_dp() . . . 904
17.91.2.2 read_spatial_data_ascii_i4() . . . 904
17.92 mo_corr::realft Interface Reference . . . 905
17.92.1 Member Function/Subroutine Documentation . . . 905
17.92.1.1 realft_dp() . . . 905
17.92.1.2 realft_sp() . . . 906
17.93 mo_orderpack::refpar Interface Reference . . . 906
17.93.1 Member Function/Subroutine Documentation . . . 906
17.93.1.1 d_refpar() . . . 906
17.93.1.2 i_refpar() . . . 906
17.93.1.3 r_refpar() . . . 906
17.94 mo_orderpack::refsor Interface Reference . . . 907
17.94.1 Member Function/Subroutine Documentation . . . 907
17.94.1.1 d_refsor() . . . 907
17.94.1.2 i_refsor() . . . 907
17.94.1.3 r_refsor() . . . 907
17.95 mo_orderpack::rinpar Interface Reference . . . 907
17.95.1 Member Function/Subroutine Documentation . . . 907
17.95.1.1 d_rinpar() . . . 907
17.95.1.2 i_rinpar() . . . 908
17.95.1.3 r_rinpar() . . . 908
17.96 mo_errormeasures::rmse Interface Reference . . . 908
17.96.1 Member Function/Subroutine Documentation . . . 908
17.96.1.1 rmse_dp_1d() . . . 908
17.96.1.2 rmse_dp_2d() . . . 908

17.96.1.3 rmse_dp_3d() . . . 909
17.96.1.4 rmse_sp_1d() . . . 909
17.96.1.5 rmse_sp_2d() . . . 909
17.96.1.6 rmse_sp_3d() . . . 909
17.97 mo_orderpack::rnkpar Interface Reference . . . 909
17.97.1 Member Function/Subroutine Documentation . . . 909
17.97.1.1 d_rnkpar() . . . 910
17.97.1.2 i_rnkpar() . . . 910
17.97.1.3 r_rnkpar() . . . 910
17.98 mo_errormeasures::sae Interface Reference . . . 910
17.98.1 Member Function/Subroutine Documentation . . . 910
17.98.1.1 sae_dp_1d() . . . 910
17.98.1.2 sae_dp_2d() . . . 911
17.98.1.3 sae_dp_3d() . . . 911
17.98.1.4 sae_sp_1d() . . . 911
17.98.1.5 sae_sp_2d() . . . 911
17.98.1.6 sae_sp_3d() . . . 911
17.99 mo_julian::setcalendar Interface Reference . . . 911
17.99.1 Member Function/Subroutine Documentation . . . 912
17.99.1.1 setcalendarinteger() . . . 912
17.99.1.2 setcalendarstring() . . . 912
17.100 mo_moment::skewness Interface Reference . . . 912
17.100.1 Member Function/Subroutine Documentation . . . 913
17.100.1.1 skewness_dp() . . . 913
17.100.1.2 skewness_sp() . . . 913
17.101 mo_mpr_global_variables::soiltype Type Reference . . . 913
17.101.1 Member Data Documentation . . . 914
17.101.1.1 clay . . . 914
17.101.1.2 db . . . 914
17.101.1.3 dbm . . . 914
17.101.1.4 depth . . . 914
17.101.1.5 id . . . 914
17.101.1.6 is_present . . . 915
17.101.1.7 ks . . . 915
17.101.1.8 nd . . . 915
17.101.1.9 nhorizons . . . 915
17.101.1.10 ntillhorizons . . . 915
17.101.1.11 z_depth . . . 915
17.101.1.12 sand . . . 915
17.101.1.13 thetafc . . . 915

17.101.1.14 thetafc_till . . . 915
17.101.1.15 thetapw . . . 916
17.101.1.16 thetapw_till . . . 916
17.101.1.17 thetas . . . 916
17.101.1.18 thetas_till . . . 916
17.101.1.19 td . . . 916
17.101.1.20 wd . . . 916
17.102 mo_orderpack::sort Interface Reference . . . 916
17.102.1 Detailed Description . . . 916
17.102.2 Member Function/Subroutine Documentation . . . 919
17.102.2.1 d_refsor() . . . 919
17.102.2.2 i_refsor() . . . 919
17.102.2.3 r_refsor() . . . 919
17.103 mo_orderpack::sort_index Interface Reference . . . 919
17.103.1 Member Function/Subroutine Documentation . . . 919
17.103.1.1 sort_index_dp() . . . 920
17.103.1.2 sort_index_i4() . . . 920
17.103.1.3 sort_index_sp() . . . 920
17.104 mo_spatial_agg_disagg_forcing::spatial_aggregation Interface Reference . . . 920
17.104.1 Detailed Description . . . 920
17.104.2 Member Function/Subroutine Documentation . . . 920
17.104.2.1 spatial_aggregation_3d() . . . 921
17.104.2.2 spatial_aggregation_4d() . . . 921
17.105 mo_spatial_agg_disagg_forcing::spatial_disaggregation Interface Reference . . . 921
17.105.1 Detailed Description . . . 921
17.105.2 Member Function/Subroutine Documentation . . . 922
17.105.2.1 spatial_disaggregation_3d() . . . 922
17.105.2.2 spatial_disaggregation_4d() . . . 922
17.106 mo_utils::special_value Interface Reference . . . 922
17.106.1 Detailed Description . . . 922
17.106.2 Member Function/Subroutine Documentation . . . 923
17.106.2.1 special_value_dp() . . . 923
17.106.2.2 special_value_sp() . . . 924
17.107 mo_kind::sprs2_dp Type Reference . . . 924
17.107.1 Detailed Description . . . 924
17.107.2 Member Data Documentation . . . 924
17.107.2.1 irow . . . 924
17.107.2.2 jcol . . . 925
17.107.2.3 len . . . 925
17.107.2.4 n . . . 925

17.107.2.5 val . . . 925
17.108 mo_kind::sprs2_sp Type Reference . . . 925
17.108.1 Detailed Description . . . 926
17.108.2 Member Data Documentation . . . 926
17.108.2.1 irow . . . 926
17.108.2.2 jcol . . . 926
17.108.2.3 len . . . 926
17.108.2.4 n . . . 926
17.108.2.5 val . . . 926
17.109 mo_errormeasures::sse Interface Reference . . . 926
17.109.1 Member Function/Subroutine Documentation . . . 927
17.109.1.1 sse_dp_1d() . . . 927
17.109.1.2 sse_dp_2d() . . . 927
17.109.1.3 sse_dp_3d() . . . 927
17.109.1.4 sse_sp_1d() . . . 927
17.109.1.5 sse_sp_2d() . . . 927
17.109.1.6 sse_sp_3d() . . . 927
17.110 mo_standard_score::standard_score Interface Reference . . . 928
17.110.1 Detailed Description . . . 928
17.110.2 Member Function/Subroutine Documentation . . . 929
17.110.2.1 standard_score_dp() . . . 929
17.110.2.2 standard_score_sp() . . . 929
17.111 mo_moment::stddev Interface Reference . . . 929
17.111.1 Member Function/Subroutine Documentation . . . 929
17.111.1.1 stddev_dp() . . . 929
17.111.1.2 stddev_sp() . . . 929
17.112 mo_corr::swap Interface Reference . . . 930
17.112.1 Member Function/Subroutine Documentation . . . 930
17.112.1.1 swap_1d_dpc() . . . 930
17.112.1.2 swap_1d_spc() . . . 930
17.113 mo_utils::swap Interface Reference . . . 930
17.113.1 Detailed Description . . . 930
17.113.2 Member Function/Subroutine Documentation . . . 931
17.113.2.1 swap_vec_dp() . . . 931
17.113.2.2 swap_vec_i4() . . . 931
17.113.2.3 swap_vec_sp() . . . 931
17.113.2.4 swap_xy_dp() . . . 931
17.113.2.5 swap_xy_i4() . . . 932
17.113.2.6 swap_xy_sp() . . . 932
17.114 mo_global_variables::twsstructure Type Reference . . . 932

17.114.2 Member Data Documentation . . . 932
17.114.2.1 basinid . . . 932
17.114.2.2 name . . . 933
17.114.2.3 ws . . . 933
17.115 mo_orderpack::uniinv Interface Reference . . . 933
17.115.1 Member Function/Subroutine Documentation . . . 933
17.115.1.1 d_uniinv() . . . 933
17.115.1.2 i_uniinv() . . . 933
17.115.1.3 r_uniinv() . . . 933
17.116 mo_orderpack::unipar Interface Reference . . . 934
17.116.1 Member Function/Subroutine Documentation . . . 934
17.116.1.1 d_unipar() . . . 934
17.116.1.2 i_unipar() . . . 934
17.116.1.3 r_unipar() . . . 934
17.117 mo_orderpack::unirnk Interface Reference . . . 934
17.117.1 Member Function/Subroutine Documentation . . . 934
17.117.1.1 d_unirnk() . . . 935
17.117.1.2 i_unirnk() . . . 935
17.117.1.3 r_unirnk() . . . 935
17.118 mo_orderpack::unista Interface Reference . . . 935
17.118.1 Member Function/Subroutine Documentation . . . 935
17.118.1.1 d_unista() . . . 935
17.118.1.2 i_unista() . . . 935
17.118.1.3 r_unista() . . . 936
17.119 mo_restart::unpack_field_and_write Interface Reference . . . 936
17.119.1 Detailed Description . . . 936
17.119.2 Member Function/Subroutine Documentation . . . 936
17.119.2.1 unpack_field_and_write_1d_dp() . . . 936
17.119.2.2 unpack_field_and_write_1d_i4() . . . 937
17.119.2.3 unpack_field_and_write_2d_dp() . . . 937
17.119.2.4 unpack_field_and_write_3d_dp() . . . 937
17.120 mo_mpr_restart::unpack_field_and_write Interface Reference . . . 937
17.120.1 Detailed Description . . . 938
17.120.2 Member Function/Subroutine Documentation . . . 938
17.120.2.1 unpack_field_and_write_1d_dp() . . . 938
17.120.2.2 unpack_field_and_write_1d_i4() . . . 938
17.120.2.3 unpack_field_and_write_2d_dp() . . . 939
17.120.2.4 unpack_field_and_write_3d_dp() . . . 939
17.121 mo_orderpack::valmed Interface Reference . . . 939
17.121.1 Member Function/Subroutine Documentation . . . 939

17.121.1.1d_valmed() . . . 939
17.121.1.2i_valmed() . . . 940
17.121.1.3r_valmed() . . . 940
17.122mo_orderpack::valnth Interface Reference . . . 940
17.122.1Member Function/Subroutine Documentation . . . 940
17.122.1.1d_valnth() . . . 940
17.122.1.2i_valnth() . . . 940
17.122.1.3r_valnth() . . . 940
17.123mo_ncwrite::var2nc Interface Reference . . . 941
17.123.1Detailed Description . . . 941
17.123.2Member Function/Subroutine Documentation . . . 942
17.123.2.1var2nc_1d_dp() . . . 942
17.123.2.2var2nc_1d_i4() . . . 943
17.123.2.3var2nc_1d_sp() . . . 943
17.123.2.4var2nc_2d_dp() . . . 943
17.123.2.5var2nc_2d_i4() . . . 944
17.123.2.6var2nc_2d_sp() . . . 944
17.123.2.7var2nc_3d_dp() . . . 944
17.123.2.8var2nc_3d_i4() . . . 944
17.123.2.9var2nc_3d_sp() . . . 945
17.123.2.10var2nc_4d_dp() . . . 945
17.123.2.11var2nc_4d_i4() . . . 945
17.123.2.12var2nc_4d_sp() . . . 946
17.123.2.13var2nc_5d_dp() . . . 946
17.123.2.14var2nc_5d_i4() . . . 946
17.123.2.15var2nc_5d_sp() . . . 947
17.124mo_ncwrite::variable Type Reference . . . 948
17.124.1Member Data Documentation . . . 949
17.124.1.1att . . . 949
17.124.1.2count . . . 949
17.124.1.3dimids . . . 949
17.124.1.4dimtypes . . . 950
17.124.1.5g0_b . . . 950
17.124.1.6g0_d . . . 950
17.124.1.7g0_f . . . 950
17.124.1.8g0_i . . . 950
17.124.1.9g1_b . . . 950
17.124.1.10g1_d . . . 950
17.124.1.11g1_f . . . 950
17.124.1.12g1_i . . . 950

17.124.1.162_b . . . 951
17.124.1.162_d . . . 951
17.124.1.162_f . . . 951
17.124.1.162_i . . . 951
17.124.1.173_b . . . 951
17.124.1.183_d . . . 951
17.124.1.193_f . . . 951
17.124.1.203_i . . . 951
17.124.1.214_b . . . 951
17.124.1.224_d . . . 952
17.124.1.234_f . . . 952
17.124.1.244_i . . . 952
17.124.1.25same . . . 952
17.124.1.26att . . . 952
17.124.1.27dims . . . 952
17.124.1.28vls . . . 952
17.124.1.29subs . . . 952
17.124.1.30start . . . 952
17.124.1.31unlimited . . . 953
17.124.1.32arid . . . 953
17.124.1.33flag . . . 953
17.124.1.34type . . . 953
17.125mo_moment::variance Interface Reference . . . 953
17.125.1Member Function/Subroutine Documentation . . . 953
17.125.1.1variance_dp() . . . 953
17.125.1.2variance_sp() . . . 953
17.126mo_errormeasures::wnse Interface Reference . . . 954
17.126.1Member Function/Subroutine Documentation . . . 954
17.126.1.1wnse_dp_1d() . . . 954
17.126.1.2wnse_dp_2d() . . . 954
17.126.1.3wnse_dp_3d() . . . 954
17.126.1.4wnse_sp_1d() . . . 954
17.126.1.5wnse_sp_2d() . . . 954
17.126.1.6wnse_sp_3d() . . . 955
17.127mo_xor4096::xor4096 Interface Reference . . . 955
17.127.1Member Function/Subroutine Documentation . . . 955
17.127.1.1xor4096d_0d() . . . 955
17.127.1.2xor4096d_1d() . . . 955
17.127.1.3xor4096f_0d() . . . 956
17.127.1.4xor4096f_1d() . . . 956

17.127.1.5 xor4096l_0d() . . . 956
17.127.1.6 xor4096l_1d() . . . 956
17.127.1.7 xor4096s_0d() . . . 956
17.127.1.8 xor4096s_1d() . . . 956
17.128 mo_xor4096::xor4096g Interface Reference . . . 957
17.128.1 Member Function/Subroutine Documentation . . . 957
17.128.1.1 xor4096gd_0d() . . . 957
17.128.1.2 xor4096gd_1d() . . . 957
17.128.1.3 xor4096gf_0d() . . . 957
17.128.1.4 xor4096gf_1d() . . . 957

18 File Documentation **959**

18.1 1-main.dox File Reference . . . 959
18.2 2-get_started.dox File Reference . . . 959
18.3 3-data_preparation.dox File Reference . . . 959
18.4 4-visualise_out.dox File Reference . . . 959
18.5 5-calibration.dox File Reference . . . 959
18.6 6-style_guide.dox File Reference . . . 959
18.7 7-test_basin.dox File Reference . . . 959
18.8 8-protocols_for_setup_new_mHM_basin.dox File Reference . . . 959
18.9 9-mRM.dox File Reference . . . 959
18.10 DEPENDENCIES.md File Reference . . . 959
18.11 mhm_driver.f90 File Reference . . . 959
18.11.1 Function/Subroutine Documentation . . . 959
18.11.1.1 mhm_driver() . . . 960
18.12 mhm_papers.md File Reference . . . 962
18.13 mo_anneal.f90 File Reference . . . 962
18.14 mo_append.f90 File Reference . . . 963
18.15 mo_canopy_interc.f90 File Reference . . . 964
18.16 mo_common_constants.f90 File Reference . . . 964
18.17 mo_common_file.f90 File Reference . . . 965
18.18 mo_common_functions.f90 File Reference . . . 965
18.19 mo_common_mHM_mRM_file.f90 File Reference . . . 966
18.20 mo_common_mHM_mRM_read_config.f90 File Reference . . . 966
18.21 mo_common_mHM_mRM_variables.f90 File Reference . . . 966
18.22 mo_common_read_config.f90 File Reference . . . 967
18.23 mo_common_read_data.f90 File Reference . . . 967
18.24 mo_common_restart.f90 File Reference . . . 968
18.25 mo_common_variables.f90 File Reference . . . 968
18.26 mo_constants.f90 File Reference . . . 969

18.27mo_corr.f90 File Reference . . . 971
18.28mo_dds.f90 File Reference . . . 972
18.29mo_errormeasures.f90 File Reference . . . 973
18.30mo_file.f90 File Reference . . . 974
18.31mo_finish.f90 File Reference . . . 975
18.32mo_global_variables.f90 File Reference . . . 975
18.33mo_grid.f90 File Reference . . . 977
18.34mo_init_states.f90 File Reference . . . 977
18.35mo_julian.f90 File Reference . . . 978
18.36mo_kind.f90 File Reference . . . 979
18.37mo_linfit.f90 File Reference . . . 980
18.38mo_mad.f90 File Reference . . . 980
18.39mo_mcmc.f90 File Reference . . . 980
18.40mo_message.f90 File Reference . . . 981
18.41mo_meteo_forcings.f90 File Reference . . . 981
18.42mo_mhm.f90 File Reference . . . 982
18.43mo_mhm_constants.f90 File Reference . . . 982
18.44mo_mhm_eval.f90 File Reference . . . 983
18.45mo_mhm_read_config.f90 File Reference . . . 983
18.46mo_moment.f90 File Reference . . . 983
18.47mo_mpr_constants.f90 File Reference . . . 985
18.48mo_mpr_eval.f90 File Reference . . . 986
18.49mo_mpr_file.f90 File Reference . . . 986
18.50mo_mpr_global_variables.f90 File Reference . . . 987
18.51mo_mpr_pet.f90 File Reference . . . 988
18.52mo_mpr_read_config.f90 File Reference . . . 989
18.53mo_mpr_restart.f90 File Reference . . . 989
18.54mo_mpr_runoff.f90 File Reference . . . 990
18.55mo_mpr_smhorizons.f90 File Reference . . . 990
18.56mo_mpr_soilmoist.f90 File Reference . . . 990
18.57mo_mpr_startup.f90 File Reference . . . 991
18.58mo_mrm_constants.f90 File Reference . . . 991
18.59mo_mrm_eval.f90 File Reference . . . 991
18.60mo_mrm_file.f90 File Reference . . . 992
18.61mo_mrm_global_variables.f90 File Reference . . . 993
18.62mo_mrm_init.f90 File Reference . . . 995
18.63mo_mrm_mpr.f90 File Reference . . . 995
18.64mo_mrm_net_startup.f90 File Reference . . . 995
18.64.1 Function/Subroutine Documentation . . . 996
18.64.1.1 calculate_l11_flow_accumulation() . . . 996

18.65mo_mrm_objective_function_runoff.f90 File Reference . . . 997
18.66mo_mrm_read_config.f90 File Reference . . . 998
18.67mo_mrm_read_data.f90 File Reference . . . 998
18.68mo_mrm_restart.f90 File Reference . . . 999
18.69mo_mrm_river_head.f90 File Reference . . . 999
18.70mo_mrm_routing.f90 File Reference . . . 1000
18.71mo_mrm_signatures.f90 File Reference . . . 1000
18.72mo_mrm_write.f90 File Reference . . . 1001
18.73mo_mrm_write_fluxes_states.f90 File Reference . . . 1001
18.74mo_multi_param_reg.f90 File Reference . . . 1002
18.75mo_ncread.f90 File Reference . . . 1003
18.76mo_ncwrite.f90 File Reference . . . 1004
18.77mo_netcdf.f90 File Reference . . . 1005
18.78mo_neutrons.f90 File Reference . . . 1008
18.79mo_nml.f90 File Reference . . . 1009
18.80mo_objective_function.f90 File Reference . . . 1009
18.81mo_optimization.f90 File Reference . . . 1010
18.82mo_optimization_utils.f90 File Reference . . . 1010
18.83mo_orderpack.f90 File Reference . . . 1011
18.84mo_percentile.f90 File Reference . . . 1013
18.85mo_pet.f90 File Reference . . . 1013
18.86mo_prepare_gridded_lai.f90 File Reference . . . 1014
18.87mo_read_forcing_nc.f90 File Reference . . . 1014
18.88mo_read_latlon.f90 File Reference . . . 1015
18.89mo_read_lut.f90 File Reference . . . 1015
18.90mo_read_optional_data.f90 File Reference . . . 1015
18.91mo_read_spatial_data.f90 File Reference . . . 1016
18.92mo_read_timeseries.f90 File Reference . . . 1016
18.93mo_read_wrapper.f90 File Reference . . . 1016
18.94mo_restart.f90 File Reference . . . 1017
18.95mo_runoff.f90 File Reference . . . 1017
18.96mo_sce.f90 File Reference . . . 1017
18.96.1 Function/Subroutine Documentation . . . 1018
18.96.1.1 set_optional() . . . 1018
18.96.1.2 write_best_final() . . . 1018
18.96.1.3 write_best_intermediate() . . . 1019
18.96.1.4 write_population() . . . 1019
18.96.1.5 write_termination_case() . . . 1020
18.97mo_set_netcdf_outputs.f90 File Reference . . . 1020
18.98mo_snow_accum_melt.f90 File Reference . . . 1020

18.99 mo_soil_database.f90 File Reference . . . 1020
18.100 mo_soil_moisture.f90 File Reference . . . 1021
18.101 mo_spatial_agg_disagg_forcing.f90 File Reference . . . 1021
18.102 mo_spatialsimilarity.f90 File Reference . . . 1022
18.103 mo_standard_score.f90 File Reference . . . 1022
18.104 mo_startup.f90 File Reference . . . 1022
18.105 mo_string_utils.f90 File Reference . . . 1023
18.106 mo_template.f90 File Reference . . . 1024
18.107 mo_temporal_aggregation.f90 File Reference . . . 1024
18.108 mo_temporal_disagg_forcing.f90 File Reference . . . 1025
18.109 mo_timer.f90 File Reference . . . 1025
18.110 mo_upscaling_operators.f90 File Reference . . . 1026
18.111 mo_utils.f90 File Reference . . . 1026
18.112 mo_write_ascii.f90 File Reference . . . 1027
18.113 mo_write_fluxes_states.f90 File Reference . . . 1028
18.114 mo_xor4096.f90 File Reference . . . 1029
18.115 mpr_driver.f90 File Reference . . . 1029
18.115.1 Function/Subroutine Documentation . . . 1029
18.115.1.1 mpr_driver() . . . 1030
18.116 mrm_driver.f90 File Reference . . . 1031
18.116.1 Function/Subroutine Documentation . . . 1032
18.116.1.1 mrm_driver() . . . 1032
18.117 RELEASES.md File Reference . . . 1033

Bibliography **1035**

Index **1037**

Chapter 1

Introduction to mHM

This chapter is divided in the following sections:

- Short Description
- The Grid-based mHM Model
- Model Formulation
- The Multiscale Parameter Regionalization Technique
- The Parameter Estimation Problem
- Model Calibration
- Test basin
- Protocols

1.1 Short Description

This document describes the source code for the mesoscale Hydrologic Model mHM. mHM is based on accepted hydrological conceptualizations and is able to reproduce as accurately as possible not only observed discharge hydrographs at any point within a basin but also the distribution of soil moisture among other state variables and fluxes. To achieve these goals and to ensure a reliable performance in ungauged basins, this model employs a multiscale parameter regionalization technique to obtain effective at the scale of interest.

This model is driven by daily or hourly precipitation, temperature fields that are acquired either from satellite products or from observation networks. In the latter case, the driving meteorological data has to be prepared in advance. A module for external drift Kriging is available upon request.

1.2 The Grid-based mHM Model

The mHM hydrologic model is based on numerical approximations of dominant hydrological processes that have been tested in various models: HBV [2], [11], [5] and VIC [13] . This model includes also a number of new features that will be described in the next section. In general, this model simulates the following processes: canopy interception, snow accumulation and melting, soil moisture dynamics, infiltration and surface runoff, evapotranspiration,

subsurface storage and discharge generation, deep percolation and baseflow, and discharge attenuation and flood routing (mHM). More information about the model can be found in [16] .

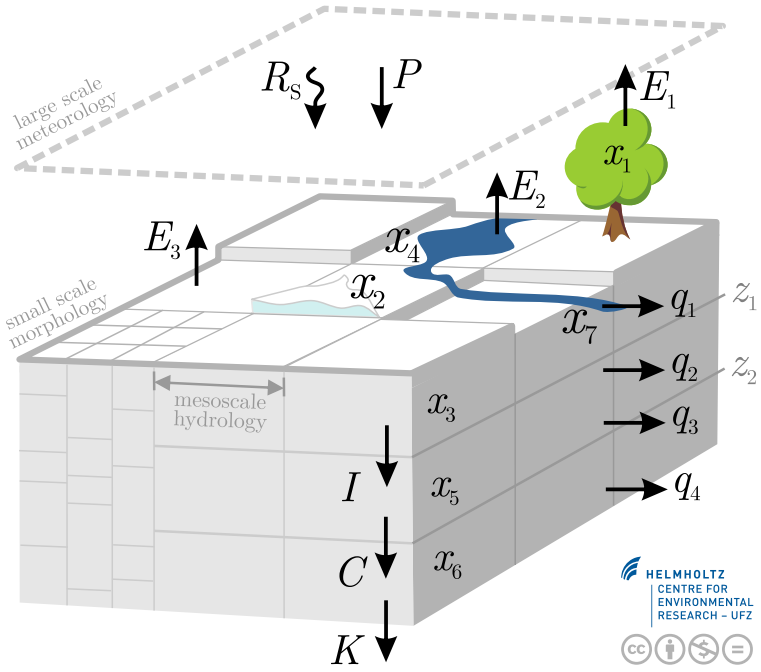


Figure 1.1: Typical mHM cell

Dominant processes of the hydrological cycle at mesoscale span over several orders of magnitude [6] . In this model, three levels (mHM levels) will be differentiated to better represent the spatial variability of state and inputs variables:

- *Level-0*: Spatial discretization suitable to describe the main features of the terrain, the main soil characteristics (pedotop), and the land cover. The cell size at this level is denoted by ℓ_0.

- *Level-1*: Spatial discretization used to describe dominant hydrological processes [4] at the mesoscale as well as the main geological formations of the basin. The cell size at this level is denoted by ℓ_1.

- *Level-2*: Spatial discretization suitable to describe the variability of the meteorological forcings at the mesoscale, for example the formation of convective precipitation. The cell size at this level is denoted by ℓ_2.

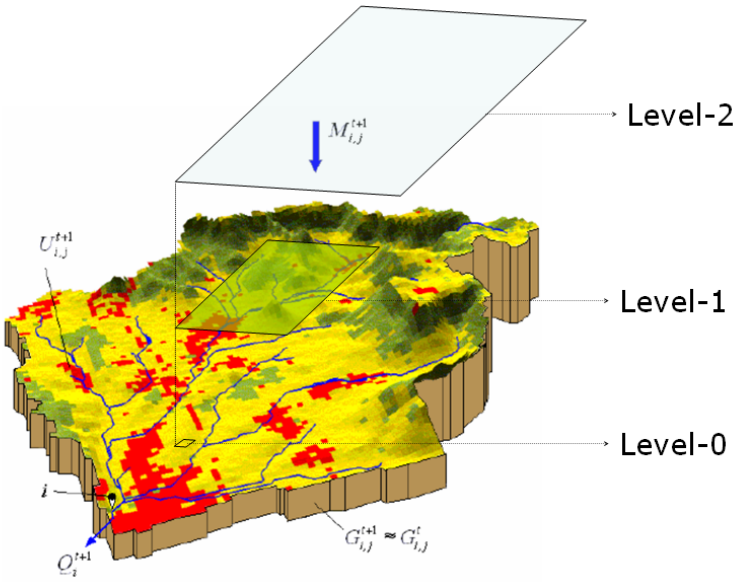


Figure 1.2: Hierarchy of data and modeling levels in mHM

1.3 Model Formulation

A mesoscale basin is an open natural system, composed of very heterogeneous materials and having fuzzy boundary conditions. The continuity assumption of the the input variables is quite difficult to justify considering that the spatial heterogeneity of a basin is mostly described by discrete attributes such soil texture types, land cover classes, and geological formations. Due to these reasons, a system of ordinary differential equations ODEs was adopted to describe the evolution of the state variables at a given location i within the domain Ω. This system of ODE is

$$
\begin{aligned}
\dot{x}_{1i} &= P_i(t) - F_i(t) - E_{1i}(t) \\
\dot{x}_{2i} &= S_i(t) - M_i(t) \\
\dot{x}^l_{3i} &= (1-\rho^l) I_i^{l-1}(t) - E^l_{3i}(t) - I^l_i(t) \\
\dot{x}_{4i} &= \rho^1 \left(R_i(t) + M_i(t) \right) - E_{2i}(t) - q_{1i}(t) \\
\dot{x}_{5i} &= I^L_i(t) - q_{2i}(t) - q_{3i}(t) - C_i(t) \\
\dot{x}_{6i} &= C_i(t) - q_{4i}(t) \\
\dot{x}_{7i} &= \hat{Q}^0_i(t) - \hat{Q}^1_i(t)
\end{aligned}
$$

$\forall i \in \Omega$.

where

Inputs	Description
P	Daily precipitation depth, $\mathrm{mm\ d^{-1}}$
E_p	Daily potential evapotranspiration (PET), $\mathrm{mm\ d^{-1}}$
T	Daily mean air temperature, $^\circ C$

Fluxes	Description
S	Snow precipitation depth, $\mathrm{mm\ d^{-1}}$
R	Rain precipitation depth, $\mathrm{mm\ d^{-1}}$
M	Melting snow depth, $\mathrm{mm\ d^{-1}}$

Fluxes	Description
E_p	Potential evapotranspiration, $\mathrm{mm\ d^{-1}}$
F	Throughfall, $\mathrm{mm\ d^{-1}}$
E_1	Actual evaporation intensity from the canopy, $\mathrm{mm\ d^{-1}}$
E_2	Actual evapotranspiration intensity, $\mathrm{mm\ d^{-1}}$
E_3	Actual evaporation from free-water bodies, $\mathrm{mm\ d^{-1}}$
I	Recharge, infiltration intensity or effective precipitation, $\mathrm{mm\ d^{-1}}$
C	Percolation, $\mathrm{mm\ d^{-1}}$
q_1	Surface runoff from impervious areas, $\mathrm{mm\ d^{-1}}$
q_2	Fast interflow, $\mathrm{mm\ d^{-1}}$
q_3	Slow interflow, $\mathrm{mm\ d^{-1}}$
q_4	Baseflow, $\mathrm{mm\ d^{-1}}$

Outputs	Description
Q_i^0	Simulated discharge entering the river stretch at cell i, $\mathrm{m^3\ s^{-1}}$
Q_i^1	Simulated discharge leaving the river stretch at cell i, $\mathrm{m^3\ s^{-1}}$

States	Description
x_1	Depth of the canopy storage, mm
x_2	Depth of the snowpack, mm
x_3	Depth of soil moisture content in the root zone, mm
x_4	Depth of impounded water in reservoirs, water bodies, or sealed areas, mm
x_5	Depth of the water storage in the subsurface reservoir, mm
x_6	Depth of the water storage in the groundwater reservoir, mm
x_7	Depth of the water storage in the channel reservoir, mm

Indices	Description
l	Index denoting a root zone horizon, $l = 1, \ldots, L$ (say $L = 3$), in the first layer, $0 \leq z \leq z_1$
t	Time index for each Δt interval
ρ^l	Overall influx fraction accounting for the impervious cover within a cell

1.4 The Multiscale Parameter Regionalization Technique

mHM requires at most 28 parameters (depending of the configuration) per cell to account for the spatial variability of the dominant hydrological processes at a mesoscale river basin. These *effective parameters* have to be estimated through calibration. Calibrating this model with a significant number of free parameters for every grid cell would lead to over-parameterization in a mesoscale catchment. This, in turn, would tend to increase the predictive uncertainty of the model due to the *equifinality* [3] of feasible solutions. Moreover, the high dimensionality of this optimization problem is also a daunting task for the state-of-the-art optimization algorithms [14] . To overcome this problem a

multiscale parameter regionalization (MPR) was employed in the mHM model [16] .

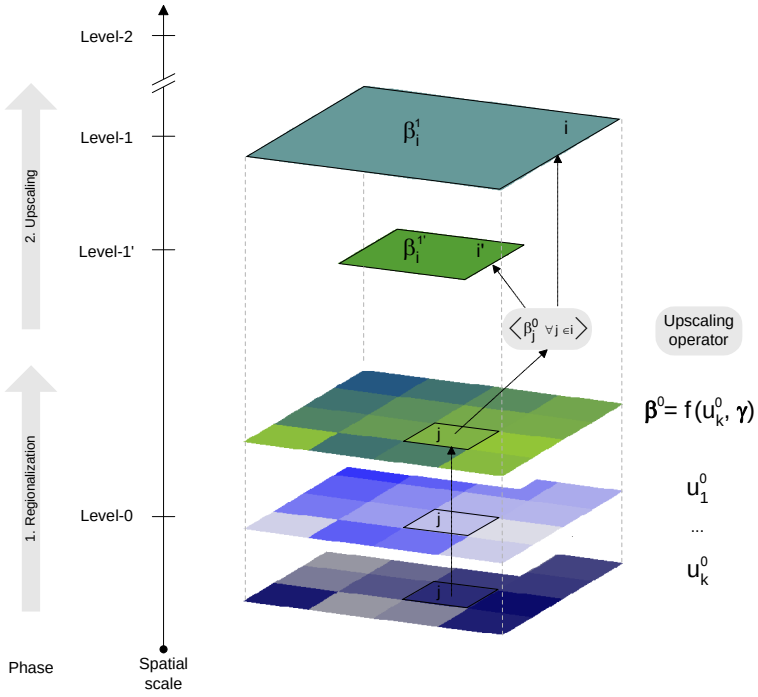


Figure 1.3: MPR

Based on this regionalization method, model parameters at a coarser grid (Level-1) are linked with their corresponding ones at a finer resolution (Level-0) (MPR). The linkage is done with upscaling operators. Model parameters at the finer scale are, in turn, regionalized with nonlinear transfer functions that couple catchment descriptors with the global parameters. Process understanding and empirical evidence are used to define these a priori relationships. The general form of an upscaling operator O is:

$$\beta_{ki}(t) = \mathrm{O}_k\left\langle \beta_{kj}(t) \quad \forall j \in i \right\rangle_i$$

where

$$\beta_{kj}(t) = f_k\Big(u_j(t), \gamma\Big).$$

Here $k = 1, \ldots, K$ with K denoting the number of distributed model parameters. u_j denotes a v-dimensional predictor vector for cell j at level-0 which is contained by cell i at level-1 (e.g. land cover, elevation, soil texture).

γ is a s-dimensional vector of transfer parameters(super parameters), with s denoting the number of free parameters to be calibrated or total degrees of freedom. $\mathrm{O}_k\langle\bullet\rangle_i$ denotes the kind of operator applied for the regionalization of the parameter k. Several types of operators were employed, e.g. majority $\mathscr{M}$, arithmetic mean $\mathscr{A}$, maximum difference $\mathscr{D}$, geometric mean $\mathscr{G}$, and harmonic mean $\mathscr{H}$. This table also shows the type of relationship employed for the transfer function and the predictors. By establishing such a relationships, the calibration algorithm finds good solutions for the transfer functions parameters ($s = 45$) instead of the model parameters for every grid cell. This, in turn, implies a great reduction of complexity since $K \times n \gg s$, where n denotes the total number of cells of a given basin at level-1.

1.5 The Parameter Estimation Problem

Let $\mathbf{M}\{\mathbf{f}, \mathbf{g}\}$ be a dynamic, spatially distributed, parameter efficient, integrated model that relates a number of state variables $\mathbf{x}$ with some *observables* called: inputs $\mathbf{u}$ and outputs $\mathbf{y}$. In general, the system is described by the following system of equations

$$\begin{aligned} \dot{\mathbf{x}}(i,t) &= \mathbf{f}(\mathbf{x},\mathbf{u},\beta,\gamma)(i,t)+\eta(i,t) \\ \mathbf{y}(i,t) &= \mathbf{g}(\mathbf{x},\mathbf{u},\beta,\gamma)_{\Omega}+\varepsilon(i,t) \end{aligned}$$

where

- $\mathbf{f}$ is a system of functional relationships in mHM (continuous or discrete) that denote the evolution of the system over time.
- $\mathbf{g}$ is a vector of functional relationships used to quantify the expected output (e.g. runoff) of the model denoted by $\hat{\mathbf{y}}$.
- ε denotes the uncertainty of the system originated by defects on measurements of both the inputs and outputs.
- η denotes the uncertainty originated by the simplification included during the formulation of $\mathbf{M}$ or due to the lack of knowledge of the relevant processes (i.e. any kind of model structure deficiency). This term is ignored in the present case.
- β denotes the fields of effective mHM parameters at level-1 estimated as described in section (MPR).
- γ is a vector of global parameters characterized by a probability density function Φ_{γ_s}.
- i, t represent a point in space and time respectively.

In general, γ can be estimated, for example, by

$$\min_{\hat{\gamma}} = \|\mathbf{y}-\hat{\mathbf{y}}\|$$

where $\|\cdot\|$ denotes a robust estimator. Many procedures to estimate global parameters are provided in mHM (e.g. simulated annealing [1], dynamically dimensioned search [17] . Other techniques can be found in the CHS Fortran Library.

1.6 Model Calibration

Good parameter sets for γ were identified with a split-sampling technique using an adaptive constrained optimization algorithm based on simulated annealing (SA) [1] . The overall model efficiency was estimated as a weighted combination of four estimators based on the Nash-Sutcliffe efficiency (NSE) between observed and calculated streamflows using three different time scales (daily, monthly and annual) as well as the logarithms of the streamflow to downplay the effects of the peak flows over the low flows [10] . These objective functions are denoted by $\phi_k, k = 1,4$. Every objective function should be normalized in the interval [0,1], with 1 representing the best posible solution. The overall objective function to be minimized is then

$$\Phi = \left(\sum_i w_i^p (1-\phi_i)^p\right)^{\frac{1}{p}}$$

where $p > 1$, and $\sum_{i=1}^{4} w_i = 1$. Here p is an exponent according to the compromise programming technique [9] and w_i denote the degree of importance of each objective. High values of p, say $p = 6$, should be chosen to avoid substitution of objective function values at low levels. In general, the estimators related to daily streamflows were twice as important as the long-term ones, thus $\{w_i\} = \{\frac{2}{4}, \frac{1}{4}, \frac{1}{4}, \frac{2}{4}\}$. The NSE for a given time interval t' is given by

$$\phi_k = 1 - \frac{\sum_{t'} \left(y_k(t') - \hat{y}_k(t')\right)^2}{\sum_{t'} \left(y_k(t') - \bar{y}_k(t')\right)^2}$$

where $\overline{y}_k(t')$ is the mean value of the observations time series over the calibration period. The index k denotes here the daily, monthly, yearly, and the transformed $\ln y(t)$ streamflow discharges. y and $\hat{y}$ denote the observed and simulated streamflows at a given time scale. Fruther details about mHM's calibration options can be found in the section Calibration Options.

1.7 Helpful links

Coding and Documentation Style (see Coding and Documentation Style)
Setup netCDF on your MacOS system (see Install NETCDF)
Data Preparation for mHM (see Data Preparation for mHM)

1.8 Test basin

mHM comes with a test basin. For details see The details about the test basin.

1.9 Protocols

To set up a new input data for mHM see Protocols for setting up a new mHM basin.

Chapter 2

Getting Started

This chapter will introduce the structure of mHM technically.

2.1 Receive mHM

As from June 2018, the current release of mHM is available through a code repository provided by the version control system GitLab (https://git.ufz.de/mhm/mhm). If you like to contribute to the code developement, external accounts be granted upon request through email to mhm-admin@ufz.de.

2.2 Install NETCDF

The mHM input and output file format is NETCDF, so the according library is required on your system. Please go to www.unidata.ucar.edu/software/netcdf in order to download the current version.

- **on Linux:** Install as described in the online documentation.
- **on MacOS:** Use the short guide below.

2.2.1 Short Guide Install to NETCDF on Linux (example Ubuntu 16.04 with gfortran v5.4.0)

The official netcdf documentation can be found under:
http://www.unidata.ucar.edu/software/netcdf/docs/getting_and_building_↩
netcdf.html

If not jet available you need to **install curl** on ypur computer. Download and unpack curl, go to the related folders and use the following commands:

```
cd /Downloads/curl-7.57
CDIR=/usr/local
./configure --prefix=${CDIR}
sudo make check
sudo make install
```

Download the five dependencies zlib, hdf5, szip, netcdf and netcdf fortran and unzip them to some folder (not the place where you will install them, e.g., /Downloads/).

Download and unpack zlib, go to the related folder and use the following commands:

```
cd /Downloads/zlib-1.2.11
ZDIR=/usr/local
./configure --prefix=${ZDIR}
make check
sudo make install
```

Download and unpack szip, go to the related folder and use the following commands:

```
cd /Downloads/szip-2.1.1
SDIR=/usr/local
./configure --prefix=${SDIR} sudo make check
sudo make install
```

Download and unpack hdf5, go to the related folder and use the following commands:

```
cd /Downloads/hdf5-1.8.19
H5DIR=/usr/local
./configure --with-zlib=${ZDIR} --prefix=${H5DIR}
make check
sudo make install
```

Download and unpack netCDF for C , which is needed for Fortran. Go to the related folder and use with the following commands:

```
cd /Downloads/netcdf-4.4.4
NCDIR=/usr/local
LD_LIBRARY_PATH=/usr/local/lib
CPPFLAGS=-I${H5DIR}/include LDFLAGS=-L${H5DIR}/lib ./configure --prefix=${NCDIR}
make check
sudo make install
```

Download and unpack netCDF for Fortran , go to the related folder and use the following commands:

```
cd /Downloads/netcdf-fortran-4.4.4
FC=gfortran CPPFLAGS=-I${H5DIR}/include LDFLAGS=-L${H5DIR}/lib ./configure --prefix=/usr/local/netcdf_4.4
      _gfortran54
make check
sudo make install
```

2.2.2 Short Guide Install to NETCDF on MacOS

Download and unpack zlib v. 1.2.11, go to the related folder and use the following commands:

```
cd /Downloads/zlib-1.2.11/
./configure --prefix=/usr/local
make
make check
make test
sudo make install
```

Download and unpack szib v. 2.1.1, go to the related folder and use the following commands:

```
cd /Downloads/szip-2.1.1/
sudo ./configure --prefix=/usr/local
F77=/usr/local/bin/gfortran ./configure --prefix=/usr/local
make
sudo make check
sudo make install
```

Download and unpack hdf5 v. 1.8.19, go to the related folder and use the following commands:

```
cd /Downloads/hdf5-1.8.19/
FC=gfortran ./configure --prefix=/usr/local --with-zlib=/usr/local --with-szlib=/usr/local --enable-
      production --enable-hl --with-pthread --with-pic
make
make check
sudo make install
```

Download and unpack netCDF for C v. 4.4.4, which is needed for Fortran. Go to the related folder and use with the following commands:

```
cd /Downloads/netcdf-4.4.4/
CPPFLAGS=-I/usr/local/include LDFLAGS=-L/usr/local/lib ./configure --prefix=/usr/local/
make
make check
sudo make install
```

Download and unpack netCDF for Fortran v. 4.4.4, go to the related folder and use the following commands:

For NAG compiler only:

```
FC=nagfor LDFLAGS='-L/usr/local/lib' CPPFLAGS='-I/usr/local/include' FCFLAGS="-fpp -mismatch_all -kind=byte
      " FFLAGS="-fpp -mismatch_all -kind=byte" ./configure --prefix=/usr/local/netcdf_4.4_nag53
```

For gfortran compiler only:

```
FC=gfortran LDFLAGS='-L/usr/local/lib' CPPFLAGS='-I/usr/local/include' ./configure --prefix=/usr/local/
      netcdf_4.4_gfortran71
```

Compile. Note that it is possible that some tests fail during the make check according to the settings of your compiler.

```
make
make check
sudo make install
```

2.3 Run mHM under CYGWIN on Windows

As from June 2019, we encourage users to make use of cmake guide:
https://git.ufz.de/mhm/mhm/blob/develop/INSTALL.md

Further CYGWIN related details might be obsolete.

A NETCDF installation under Windows7 and Windows10 has only been tested using CYGWIN (https://www.cygwin.com/). When executing the CYGWIN setup, the following packages have to be installed (as of May 2018, screenshots kindly provided by Mehmet Cuneyd Demirel):

First, the following netcdf packages have to be installed (fig_cygwin_netcdf):

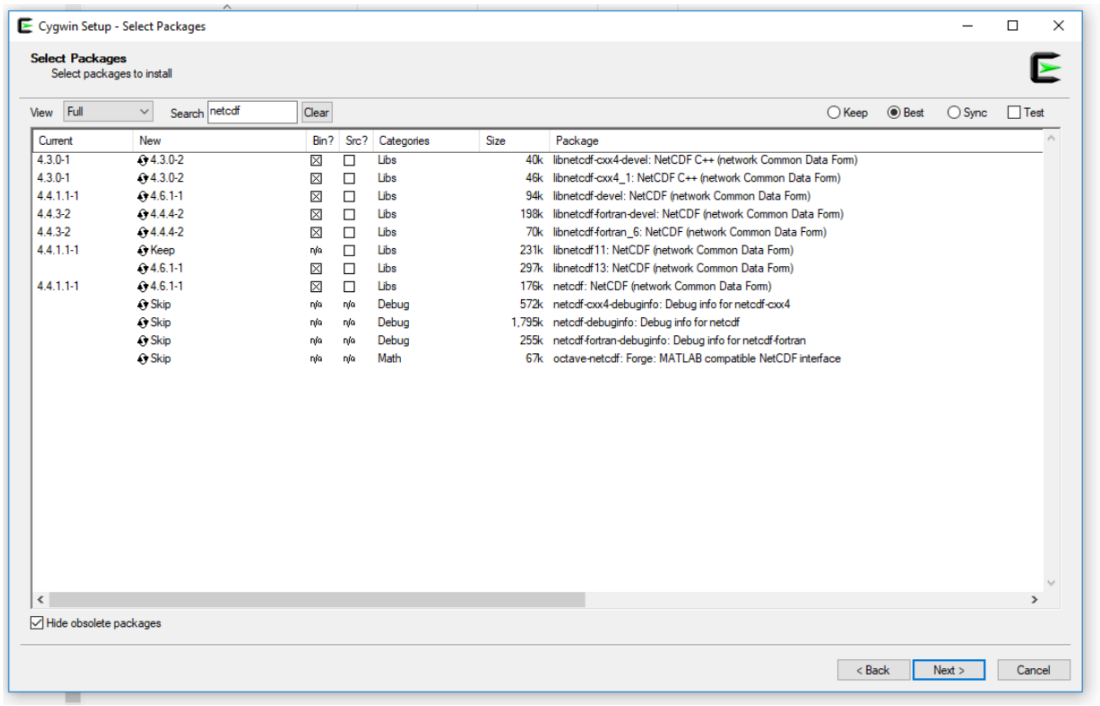


Figure 2.1: netcdf packages for CYGWIN

Second, the following hdf5 version has to be installed for netcdf-4 support.

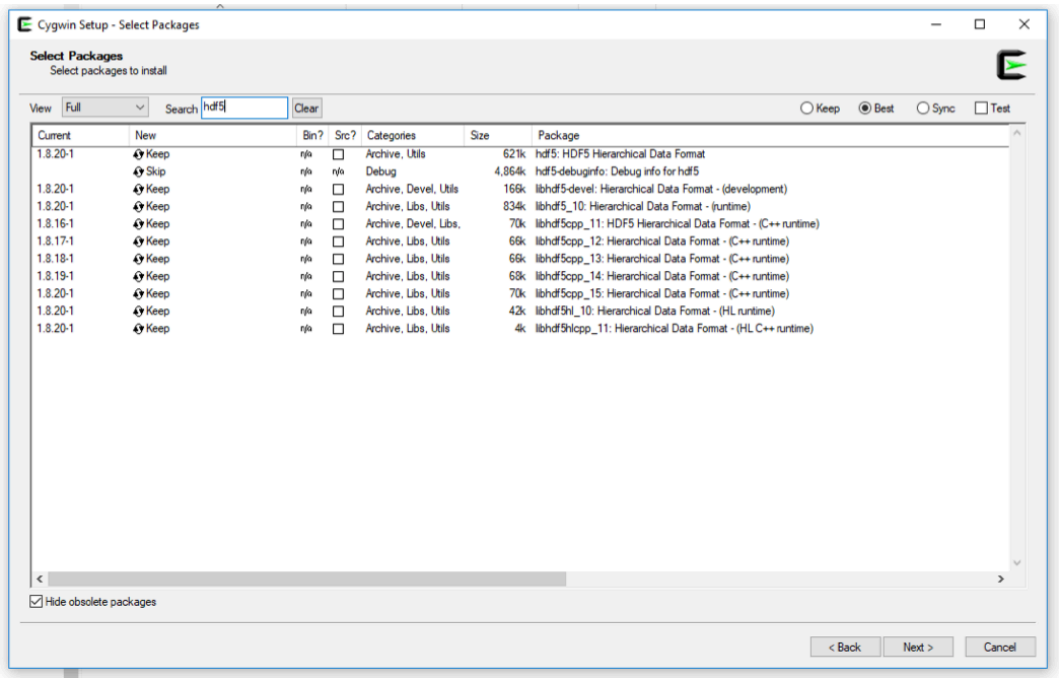


Figure 2.2: hdf5 packages for CYGWIN

Third, the gfortran compiler (fig_cygwin_gfortran) and (fig_cygwin_libgfortran) have to be installed:

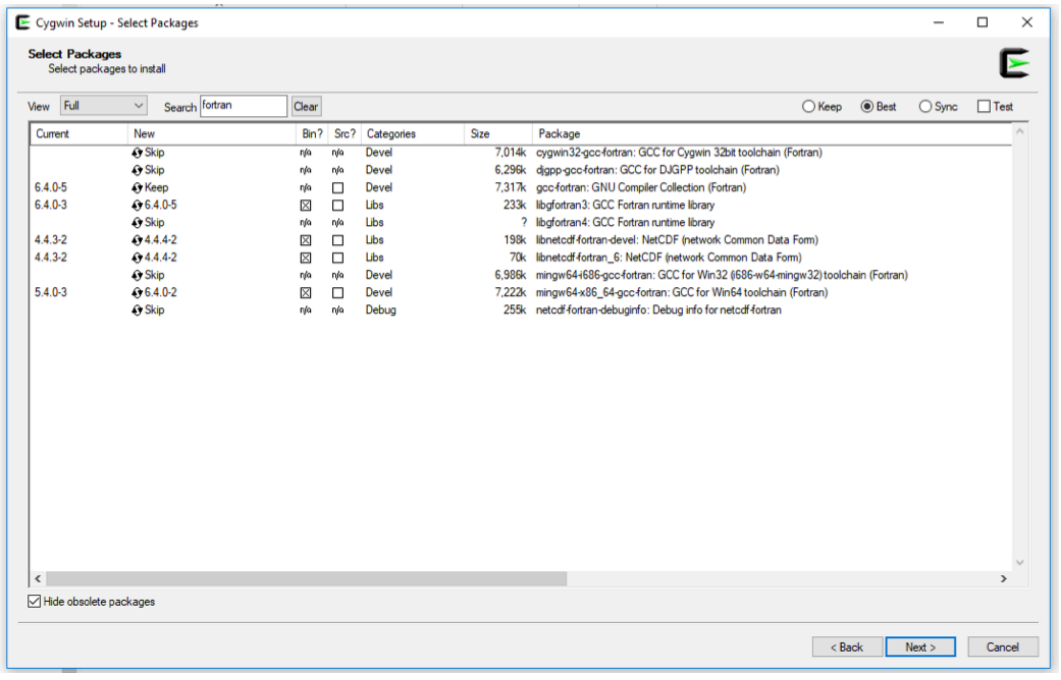


Figure 2.3: gfortran packages for CYGWIN

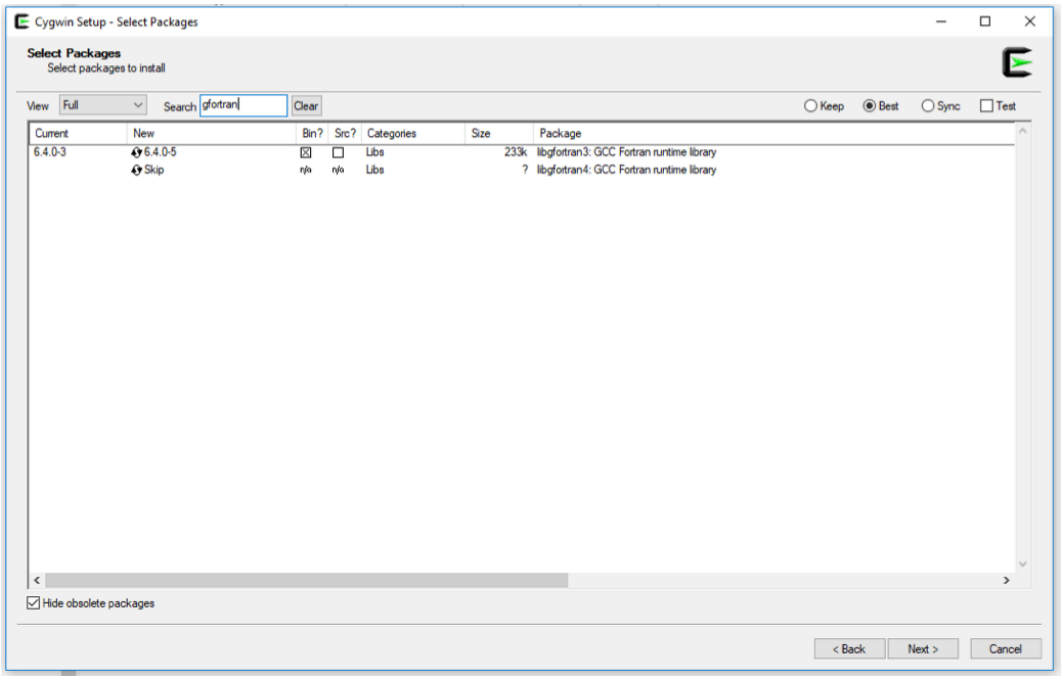


Figure 2.4: libgfortran packages for CYGWIN

Fourth, GNU `make` has to be made available by selecting the following:

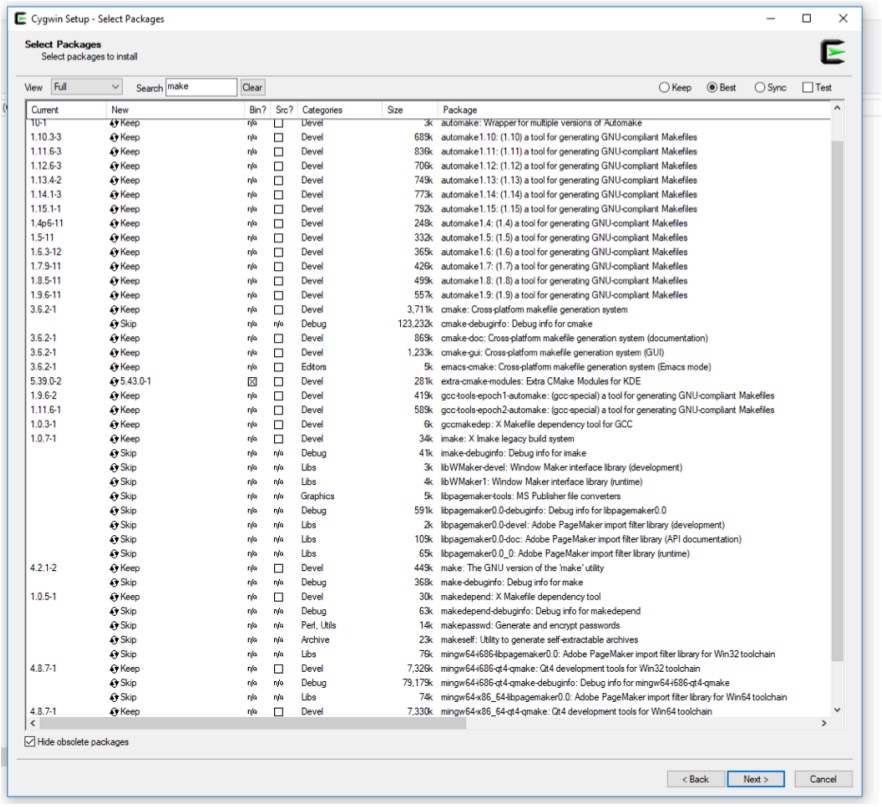


Figure 2.5: make packages for CYGWIN

Note that Cygwin users are advised to define following flag in the Makefile, otherwise compilation won't be successful.

```
EXTRA_LDFLAGS += -Wl,--stack,12485760
EXTRA_CFLAGS  += -Wl,--stack,12485760
```

Additionally, make sure to have folder in your environmental path (fig_cygwin_path).

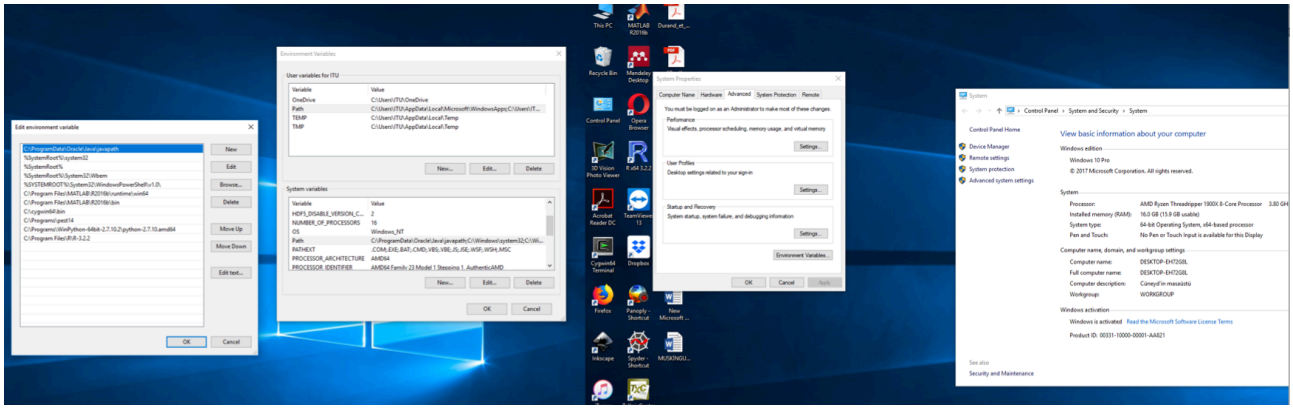


Figure 2.6: setup environmental path CYGWIN

Finally, once the installation of CYGWIN including the packages outlined above is completed, you have to change the system in the Makefile to 'cygwin'. Then, the mhm has proven to compile (simply type `make` on the command line) under Windows. This will produce an executable `mhm` which can be launched by `./mhm` and results for the test basins can be found under `./test_basin/output_b1/` and `./test_basin_2/output_b1/`.

2.4 Compile mHM

Compilations of the code need to know paths and locations of its dependencies that are specific to the individual operating system. Before compiling, make sure that you individual requirements hold.

Open `Makefile` and adjust at least the following settings:

- `system := [SYSNAME]`
  Choose an arbitrary name of your system and add your settings in `make.config/[SYSNAME].alias`. For assistance you can adapt the existing profiles in `make.config/`.
- `compiler := [COMPILER]`
  Choose a compiler that was defined in `make.config/[SYSNAME].[COMPILER]`.
- `release := debug|release`
  Debug mode checks all dependencies and code fractions and will we slower during compilation.
- `netcdf := netcdf4`
  Choose the netcdf version your system is using.

Additional assistance for editing Makefiles are available throughout the internet. Finally, mHM can be compiled with the simple command

```
make system=cygwin compiler=gnu release=debug
```

In between two compilations, it might be useful to apply the command `make cleanclean` in order to start the next `make` from scratch (without using old temporary files).

2.5 Test mHM on Example Basin

The mHM distribution usually comes with an example test basin located in

```
test_basin/
```

The directory provides input data for the test basin and output examples. Detailled information about this test basin can be found in the chapter The details about the test basin. All parameters and paths are already set by default to the test case, so you can just start the simulation with the command

```
./mhm
```

This will run mHM on two basins simultanously and create output files for discharge and interception in `test/output_b*`. The chapter Visualizing Model Output provides further information on visualising mHM results.

2.6 Run your own Simulation

Pretty much the first step mHM takes during runtime is reading the three configuration files:

- `mhm.nml`
- `mhm_output.nml`
- `mhm_parameters.nml`

When editing these files, we recommend to use syntax highlighting for Fortran, which is featured in emacs, for example.

2.6.1 Main Configuration: Paths, Periods, Switches

The file `mhm.nml` contains the main configuration for running mHM in your catchments. Since the comments should explain each single setting, this section will only roughly describe the structure of the file. By the way, paths can be relative, absolute and even symbolic links.

- **Common Settings:** Defines output path, input look-up tables and input data format (all "nc" or all "bin") for all basins in your simulation.
- **Basin wise paths:** Set paths for input and output. Create a block for each basin. Remove needless blocks.
- **Resolution:** Hourly or Daily time step. Hydrologic resolution should be factorisable with the input resolutions (e.g. meteo: 4000, hydro: 2000, morph: 100). The routing resolutions determines the velocity of water from cell to cell (keep it greater than the hydro resolution). If you change the routing, remember to recalibrate the model (see Calibration and Optimization).
- **Restart:** mHM does provide you the option to save the whole model configuration (incl. states, fluxes, routing network, and model parameters) at the end of the simulation period. mHM is then able to restart a new simulation with this model configuration. This reduces the amount of computational time in the newly started simulation because mHM does not have to re-setup the model configuration (e.g., parameter fields and routing network).
- **Periods:** Your actual period of interest should be subsequent of a warming period, where model dynamics can set in properly. Remember to expand your input data by that period, too.
- **Soil Layers:** Soil parameters (not properties) are upscaled vertically in mHM. In this section you specify the number of horizons and depths for the hydrological processing. (This for example you do not find in any other model)

- **Land Cover:** You can provide different land cover files for different years.
- **LAI data:** Switch for choosing LAI option. The user can either select to run mHM with monthly look-up-table LAI values defined for 10 land classes or choose to run mHM using gridded LAI input data (e.g., MODIS). Gridded LAI data must be provided on a daily time-step at the Level-0 resolution. /∗!
- **Process Switches:**
  - Proccess case 5 - potential evapotranspiration (PET):
    1. PET is input (processCase(5)=0)
    2. PET after Hargreaves-Samani (processCase(5)=1)
    3. PET after Priestley-Taylor (processCase(5)=2)
    4. PET after Penman-Monteith (processCase(5)=3)
  - Proccess case 8 - routing can be activated (=1) or deactivated (=0).
- **Annual Cycles:** Values for pan evaporation hold in impervious regions only. The meteorological forcing table disaggregates daily input data to hourly values.

2.6.2 Output Configuration: Time Steps, States, Fluxes

The file `mhm_output.nml` regulates how often (e.g. `timeStep_model_outputs = 24`) and which variables (fluxes and states) should be written to the final netcdf file `[OUTPUT_DIRECTORY]/mHM_Fluxes_↩States.nc`. We recommend to only switch on variables that are of actual interest, because the file size will greatly increase with the number of containing variables. During calibration (see Calibration and Optimization) no output file will be written.

2.6.3 Regionalised Parameters: Initial Values and Ranges

The file `mhm_parameters.nml` contains all global parameters and their initial values. They have been determined by calibration in German basins and seem to be transferabel to other catchments. If you come up with a very different catchment or routing resolution, these parameters should be recalibrated (see section Calibration and Optimization).

2.7 Calibration and Optimization

By default, mHM runs without caring about observed discharge data. It will use default values of the global regionalised parameters , defined in `mhm_parameters.nml` . In order to fit discharge to observed data, mHM has to be recalibrated. This will optimise the parameters such that mHM arrives at a best fit for discharge. The optimization procedure runs mHM many many times, sampling parameters in their given ranges for each iteration, until the objective function converges to a confident best fit.

2.7.1 The Optimization Routines

mHM comes with four available optimization methods:

- **MCMC:** The Monte Carlo Markov Chain sampling of parameter sets is recommended for estimation of parameter uncertainties. Intermediate results are written to `mcmc_tmp_parasets.nc` .
- **DDS:** The Dynamically Dimensioned Search is an optimization routine known improve the objective within a small number of iterations. However, the result of DDS is not neccessarily close to your global optimum. Intermediate results are written to `dds_results.out` .
- **Simulated Annealing:** Simulated Annealing is a global optimization algorithm. SA is known to require a large number of iterations before convergence (e.g. 100 times more than DDS), but finds parameter sets closer to the global minimum like DDS. Intermediate results are written to `anneal_results.out` .

- **SCE:** The Shuffled Complex Evolution is a global optimization algorithm which is based on the shuffling of parameter complexes. It needs more iterations compared to the DDS (e.g. 20 times more), but less compared to Simulated Annealing. The increasing computational effort (i.e. iterations) leads to more reliable estimation of the global optimum compared to DDS. Intermediate results are written to `sce_results.out` .

Objective functions currently implemented in mHM are:

- **NSE:** Nash-Sutcliffe Efficiency, assuming constant + linearly relative errors. Recommended for fitting high flows.
- **lnNSE:** logarithmic Nash-Sutcliffe Efficiency, recommended for fitting low flows.
- **0.5∗(NSE+lnNSE):** weights both NSE and lnNSE by 50%, roughly fits high and low flows.
- **Likelihood:** The confidence bands are probability density functions that capture variable errors, recommended for hydrological discharge.
- **KGE:** Kling Gupta model efficiency: combined measure for variability, bias and correlation
- **PD:** Pattern dissimilarity (PD) of spatially distributed soil moisture
- **ETC:** Combination of other multi-objective functions (streamflow, TWS anomaly, evapotranspiration, soil moisture), see mhm.nml for details

2.7.2 Calibration Settings for mHM

The following settings in `mhm.nml` are required for calibrating mHM:

- `optimize = .true.`
- `opti_method = 1`
  Choose methods: 0 (MCMC), 1 (DDS), 2 (SA), 3 (SCE)
- `opti_function = 1`
  Choose objective functions: 1 (NSE), 2 (lnNSE), 3 (50% NSE+lnNSE), 4 (Likelihood)
- `nIterations = 40000`
  Maximum number of iterations (mHM runs), optimisers will exit earlier if convergence criteria are reached.
- `seed = -9`
  The default value -9 will take a seed number from the system clock. Be warned that simulations might not be random if you define a positive seed here.

More specific settings are offered at the very end of the file `mhm.nml`.

In `mhm_parameters.nml` you find the initial values and ranges from which the optimiser will sample parameters. Most ranges are very sensitive and have been determined by detailed sensitivity analysis, so we do not recommend to change them. With the FLAG column you may choose which parameters are allowed to be optimised. The parameter `gain_loss_GWreservoir_karstic` is not meant to be optimised.

Please mind that optimization runs will take long and may demand a huge amount of hardware ressources. We recommend to submit those jobs to a cluster computing system.

2.7.3 Final Calibration Results

During calibration, mHM does not write out fluxes, states and discharge, so you need to perform a final run with the calibrated parameters. When the simulations are finished, the optimal parameter set is written to

```
[OUTPUT_DIRECTORY]/FinalParams.out
[OUTPUT_DIRECTORY]/FinalParams.nml
```

You can run mHM directly with the generated namelist (by renaming it) or incorporate the results in `Final↩ Params.out` as new initial values into `mhm_parameters.nml`. There are two scripts that help you with that:

- `pre-proc/create_multiple_mhm_parameter_nml.sh`
  Useful for many optimisation runs (many lines in `FinalParams.out`)
- `post-proc/opt2nml-params.pl`
  Useful to analyze where the optimisation arrived. Using two arguments, `pathto/FinalParams.out` and `pathto/mhm_parameters.nml`, it will (1) create a new `mhm_parameters.nml.opt` filled with the new values and (2) directly show the new parameters within their ranges and warn if some have been close to their end of range by 1%.

As soon as the new parameters are set, deactivate the `optimize` switch in `mhm.nml` and **rerun** mHM once in order to obtain the final optimised output.

You should also have a look at the parameter evolution (e.g. `sce_results.out`) or final results. If any of the parameters stick to the very end of their allowed range, the result is not optimal and you will be in serious trouble. Possible reasons might be bad parameter ranges (even though they have been optimised mathematically, but in a basin or resoluion not comparable to yours) or bad input data.

Chapter 3

Data Preparation for mHM

This chapter is divided in the following sections:

- Getting Started
- Preparation of the Forcings
- Preparation of the Morphological Data
- A possible GIS workflow
- Table Data
- Land Cover Data

3.1 Getting Started

To run mHM the user requires a number of datasets. The follwing subsection gives a short overview and references to freely available datasets.

3.1.1 Meteorological variables

Meteorological data is usually available from the weather services of the countries of modelling interest. Freely available alternatives are the EOBS (http://www.ecad.eu/download/ensembles/download.php) and the WATCH datasets (http://www.eu-watch.org/gfx_content/documents/README-WFD↩EI.pdf).

You may have difficulties finding measurement data for Potential Evapotranspiration (PET). One possible solution, would be to calculate PET from the much easier available variables mean, maximum and minimum air temperature, using the Hargreaves-Samani method.

The two meteorological variables which are needed:

Name	Unit	Temporal resolution
Precipitation	mm	hourly to daily
Average air temperature	°C	hourly to daily

Dependent of the specification for the potential evapotranspiration (processCase(5)) additional meteorological variables may be need:

- processCase(5) = 0 - PET is read from input.

- processCase(5) = 1 - Hargreaves-Samani equation
- processCase(5) = 2 - Priestley-Taylor equation
- processCase(5) = 3 - Penam-Monteith equation

Name	Unit	Temporal resolution
0 - Potential evapotranspiration	mm	hourly to daily
1 - Minimum air temperature	$^{\circ}\mathrm{C}$	daily
1 - Maximum air temperature	$^{\circ}\mathrm{C}$	daily
2 - Net radiation	$W\ m^{-2}$	daily
3 - Net radiation	$W\ m^{-2}$	daily
3 - Absolut vapur pressure of air	Pa	daily
3 - Windspeed	$m\ s^{-1}$	daily

3.1.2 Morphological variables

In addition to the Digital Elevation Models (DEM) usually provided by federal authorities, a number of free alternatives exists. SRTM data is available from 60° South to 60° North in a 3" ($\sim$ 90 m) resolution (http://srtm.csi.cgiar.org/), the ASTER-GDEM covers the entire globe with a resolution of 1" (http://gdem.ersdac.jspacesystems.or.jp/). Hydrographically corrected SRTM data and a number of derived products in different resolutions are available from the HydroSHEDS project (http://hydrosheds.cr.usgs.gov/index.php).

Soil and hydrogeological data is usually provided by the geological surveys. On the soil side the Harmonized World Soil Database would be a free alternative (http://webarchive.iiasa.ac.at/Research/LUC/External-World-soil-database/HTML/).

Name	Unit	Temporal resolution
Digital elevation model	m	-
Soil maps with textural properties (% sand and clay contents, bulk density per horizon, and root depth zone)	-	-
Geological maps with aquifer properties (specific yield, permeability, aquifer thickness)	-	-
Streamflow location	(m,m) (lat,lon)	-

3.1.3 Land Cover

Free land cover data is available from different sources. The Corine programm provides land cover scenes for Europe with resolutions of 100m and 250m (http://www.eea.europa.eu/publications/COR0-landcover), the Global Land Cover Map 2000 a dateset covers the entire world in a resolution of 1km.

Name	Unit	Temporal resolution
Land cover scenes	-	monthly to annual
Leaf area index	-	weekly to monthly

3.1.4 Gauging Station Information

Streamflow measurments should also be available from federal authorities. In addition, the Global Runoff Data Center provides timeseries of varying length for several thousand gauging stations all over the world (http://www.bafg.de/GRDC)

Name	Unit	Temporal resolution
Streamflow measurements	$\mathrm{m}^3\mathrm{s}^{-1}$	hourly to daily

3.1.5 Optional Data

Name	Unit	Temporal resolution
Snow cover	-	daily to weekly
Land surface temperature	K	daily to weekly
Groundwater station location	(m,m) (lat,lon)	-
Groundwater head measurements	m	weekly to monthly
Eddy covariance station location	(m,m) (lat,lon)	-
Eddy covariance measurements	m	hourly

3.2 Preparation of the Forcings

Forcing data should be available from federal databases like DWD for Germany. Be sure to bring data in the netcdf format, like

```
precipitation-1950-2013.nc
```

These files can be edited with the CDO module package in order to select or change data. Please read the according CDO and NCKS reference sheets for details and make sure to load the CDO modules before starting.

3.2.1 Extract Data to the Size of a Catchment

Let FULLMAP.nc be the the forcing data of a region much greater than your catchment. Let X1, X2, Y1, Y2 be the coordinates (lon and lat) of a rectangular overlapping the catchment. The forcing data MAP.nc for your catchment can be extracted by the CDO command:

```
cdo sellonlatbox,X1,X2,Y1,Y2 FULLMAP.nc MAP.nc
```

Optionally, certain pixels can be extracted, too:

```
ncks -d x,1,1 -d y,2,2 MAP.nc PIXEL.nc
```

3.2.2 Extact Data to the Time Period of Interest

Let MAP.nc be the forcing data including your period of interest. Let DATE1 and DATE2 be the starting and ending dates of your period, respectively. Their date format is YYYY-MM-DD. The data PERIOD.nc can be extracted by the CDO command:

```
cdo seldate,DATE1,DATE2 MAP.nc PERIOD.nc
```

Please note that a reference date according to DATE1 should be provided in PERIOD.nc:

```
cdo setreftime,DATE1,00:00:00,days PERIOD.nc FINAL.nc
```

3.2.3 Data Types

Any data provided for mHM should be of the data type DOUBLE. You should convert all data related to a variable VAR (e.g. tavg) with the following CDO command:

```
cdo -b F64 selname,VAR FINAL.nc FINAL64.nc
```

The only exception here is the time, its data type has to be INTEGER. This can be changed with NCAP2:

```
ncap2 -s 'time=int(time)' STILLNOTFINAL.nc FINAL.nc
```

3.2.4 Variable Names

In mHM the variable names of the forcing are hard-coded. They need to be

- "tavg" for average temperature
- "pre" for precipitation
- "pet" for evapotranspiration

For example, you can change the variable name TEMPERATURE in your NETCDF file with the following CDO command:

```
cdo chname,TEMPERATURE,tavg TEMPDATA.nc TEMPDATA-FINAL.nc
```

3.2.5 The Header File

Every meteorological data file should come with an additional file "header.txt" in the same directory. In the following example, we has only one pixel of data (ncols=nrows=1) and a cell size of 4km. The easting and northing coordinates of the lower left cornaer should be similar to the morphological data of your catchment.

```
ncols         1
nrows         1
xllcorner     639357.3
yllcorner     5723706.2
cellsize      4000
NODATA_value  -9999
```

3.2.6 NODATA values

In order to be consistent in your data, you should specify the same NODATA value everywhere. For example, specify "-9999" in your header file as NODATA value. Usually, this should be changed also in NC files by the `ncatted` command. The following example overwrites or adds (o) the NODATA attribute "_FillValue" with value "-9999" of type double (d) to the variable "pre":

```
ncatted -O -a _FillValue,pre,o,d,-9999. INOUT.nc
```

3.3 Preparation of the LatLon Grid

As input mHM additionally needs a `latlon.nc` file specifying the geographical location of every grid cell in WGS84 coordinates. This file has to be adjusted for every resolution of the level-1 hydrological simulations.

For creating a latlon file mHM comes with the python script `create_latlon.py`, which can be found in `pre-proc/`. Detailed information for the usage of this python script can be found in the online help by typing:

```
python [MHM_DIRECTORY]/pre-proc/create_latlon.py -h
```

`create_latlon.py` needs several specifications via command line switches. First, the coordinate system of the morphological and meteorological data have to be specified according to www.spatialreference.org by using the switch: `-c`. Second, you need to specify three header files containing the corresponding information for

the different spatail resolutions connected to mHM. The three resolutions are the resoltution of 1) the morphological input (switch: `-f`), 2) the hydrological simulation (switch: `-g`), and 3) the routing (switch: `-e`).

These header files can be produced by adopting the header file of the meteorological data Therefore, copy one of these files that you generated for meteo data (see Preparation of the Forcings):

```
cp [INPUT_DIRECTOY]/input/meteo/pre/header.txt [INPUT_DIRECTORY]/input/latlon/
```

Edit the new file `header.txt` such that `cellsize` equals your hydrologic resolution. You have to adapt `ncols` and `nrows` to that resolution such that it covers the whole region. For example, reducing the cell size from 2000 to 100, the number of columns and rows should be increased by a factor of 20.

Third, you need to set the path and filename for the resulting output file by using the switch `-o`.

An example for creating a lat-lon-file looks like

```
python [MHM_DIRECTORY]/pre-proc/create_latlon.py -c epsg:31463 -f header_100m.txt -g header_1000m.txt -e
      header_2000m.txt -o ../latlon/latlon.nc
```

Keep in mind that you need one latlon file for each hydrologic resolution in mHM. Since the filename `latlon.nc` is hard-coded in mHM, we recommend to create different directories for each resolution you need, containing the related header and latlon file.

3.4 Preparation of the Morphological Data

MHM needs several morphological input datasets. All these have to be provided as raster maps in the ArcGis ascii-format, which stores the above header and the actual data as plain text. Some of the raster files need to be complemented by look-up tables providing additional information. The tables and their structure are described in more detail in subsection Table Data. Take care of the follwing limitations to mHM input data during data processing:

- All gridded input, i.e. your morphological and your meteorological data, needs to cover the same spatial domain. That means, that the values xllcorner, yllcorner, xllcorner+ncols∗cellsize and yllcorner+nrows∗cellsize have to be identical for all files!
- MHM allows you to provide your meteorological forcing in a different horizontal resolution than the morphological data. The larger cellsize however needs to be a multiple of the smaller.

The required datasets and their corresponding filenames:

Description	Raster file name	Table file name
Sink filled Digital Elevevation Model (DEM)	dem.asc	-
Slope map	slope.asc	-
Aspect Map	aspect.asc	-
Flow Direction map	fdir.asc	-
Flow Accumulation map	facc.asc	-
Gauge(s) position map	idgauges.asc	[gauge-id].txt
Soil map	soil_class.asc	soil_classdefinition.txt
Hydrogeological map	geology_class.asc	geology_classdefinition.txt
Leaf Area Index (LAI) map	LAI_class.asc	LAI_classdefinition.txt
Land use map	your choice	-

3.5 A possible GIS workflow

In the following paragraphs a possible GIS workflow is outlined using the software ArcMAP 10 with the Spatial Analyst Extension and optionally the Arc Hydro Tools (http://downloads.esri.com/blogs/hydro/AH2/ArcHydroTools_2_0.zip)

3.5.1 General considerations

- As the spatial discretizations (i.e. resolutions, origin) of your datasets will most likely differ, it is recommended to set the following envirnmental settings on every processing step outlined in the following paragraphs.

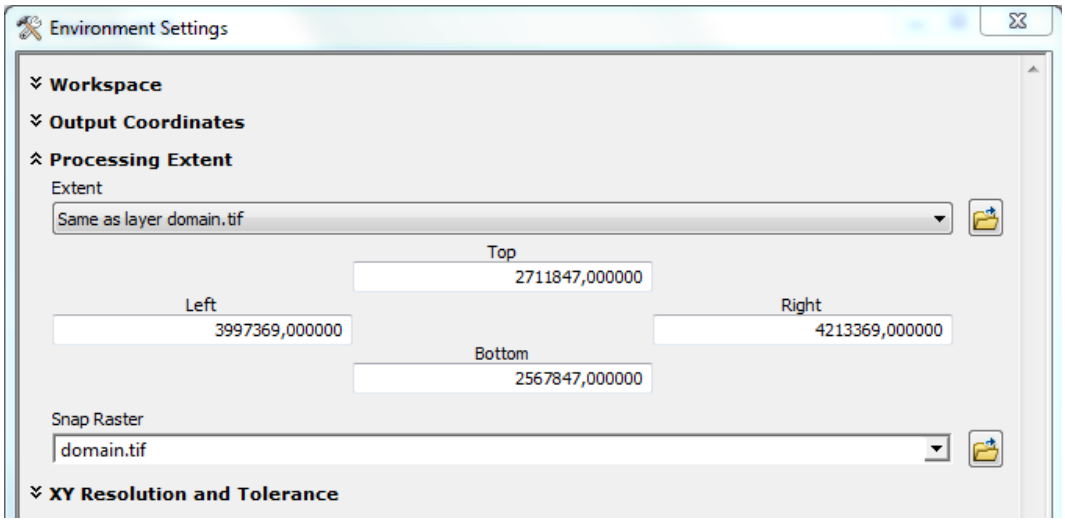


Figure 3.1: Button 'Environments...' in all Toolbox windows

Point to the largest of your input data grids in 'Extent' and 'Snap Raster', which is usually the dataset with the coarsest horizontal resolution. In the likely case, that this is your meteorological input, create a grid from any of your netcdf-files first.

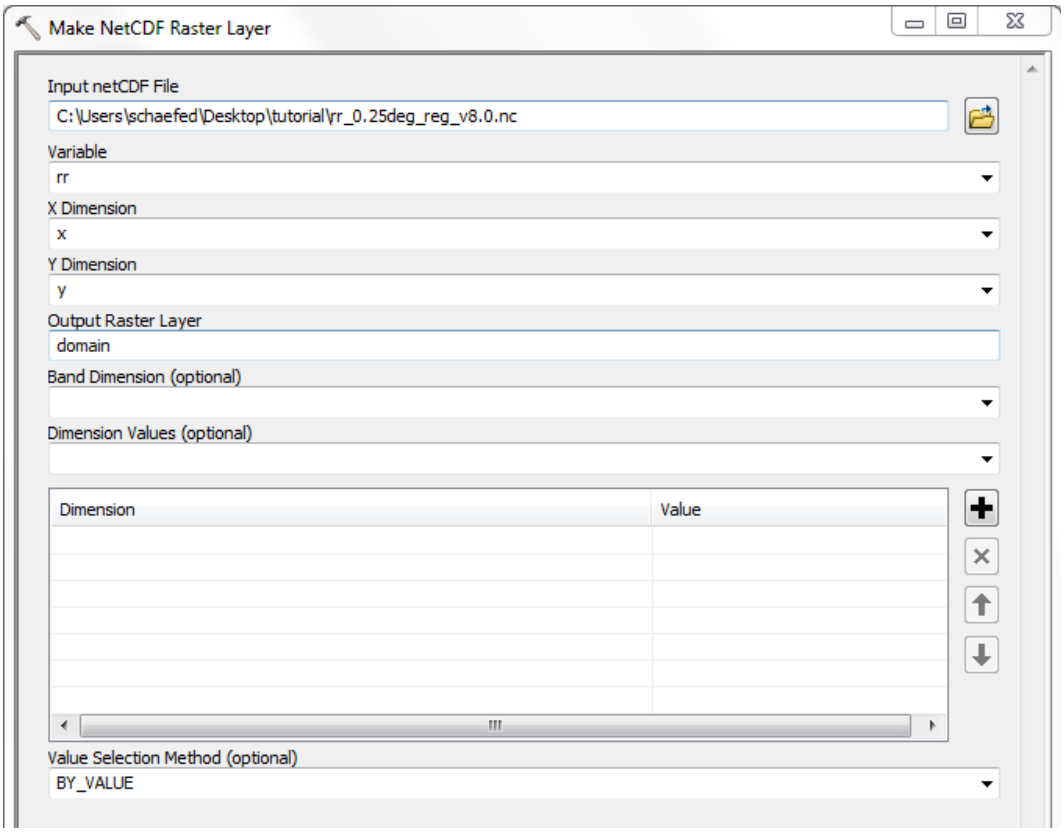


Figure 3.2: System Toolboxes -> Multidimension Tools -> Make NetCDF Raster Layer

Save the output grid (right click the raster in the 'Table of Contents' -> 'Data' -> 'Export Data...').

- If the map projections of your datasets differ, a harmonization of these becomes necessary. Repeat the depicted step to convert all your input files. Which projection to choose is highly dependent on your simulation domain and size. For the current model version an equal-area projection is strongly recommended.

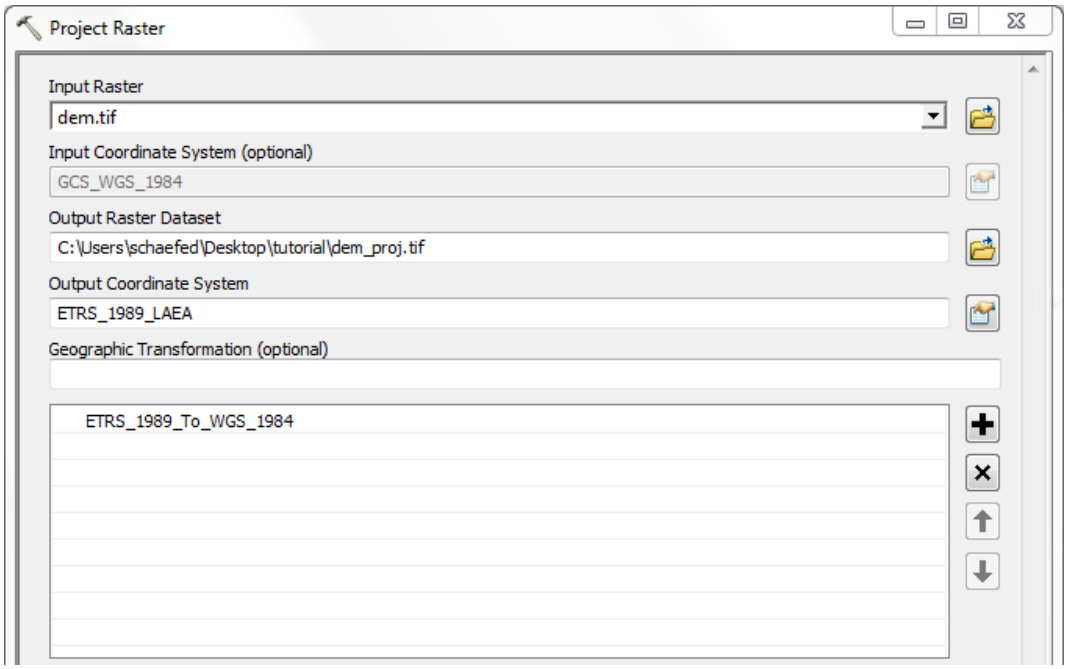


Figure 3.3: System Toolboxes -> Data Management Tools -> Projections and Transformations -> Raster -> Project Raster

- If your input datasets are not already in the desired level-0 resolution, resample the DEM, the hydrogeological, LAI, soil and land use maps. Choosing an appropiate resolution depends on data quality and needed level of simulation detail, but keep in mind that:

1. Your different input resolution levels must be multiples of each other. E.g. you should choose a level-0 resolution of 100m (instead of 90m in case you are using SRTM data) if your meteorological input resolution is 4km.
2. Model runtime directly depends on the number of grid cells.

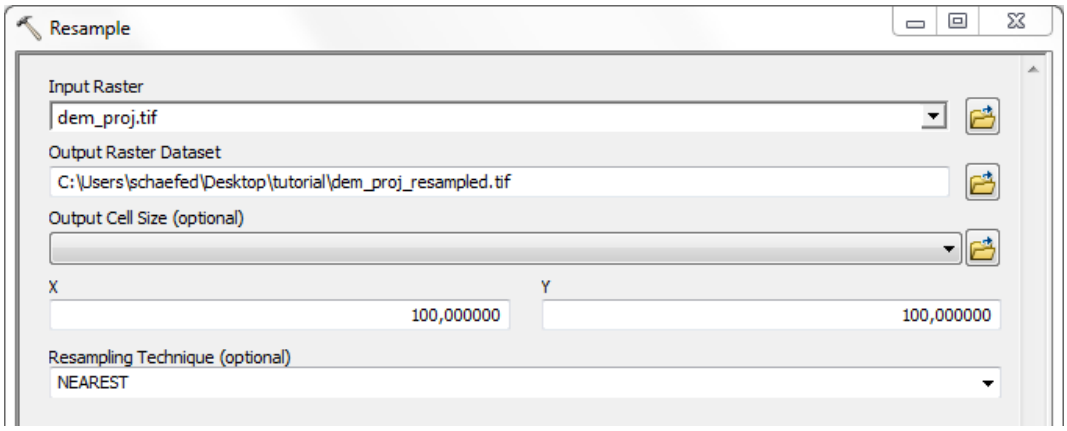


Figure 3.4: System Toolboxes -> Data Management Tools -> Raster -> Raster Processing -> Resample

- It is important that all your morphological input files exactly cover the same spatial domain. That also means that if a cell contains valid data in any one of the datasets, the very same cell must also be definied in all the others. One possibility to solve this typical problem would be to set such 'doubtful' cells to the corresponding NODATA_value. Therefore create a mask as depicted below, which only contains cells, that are defined everywhere.

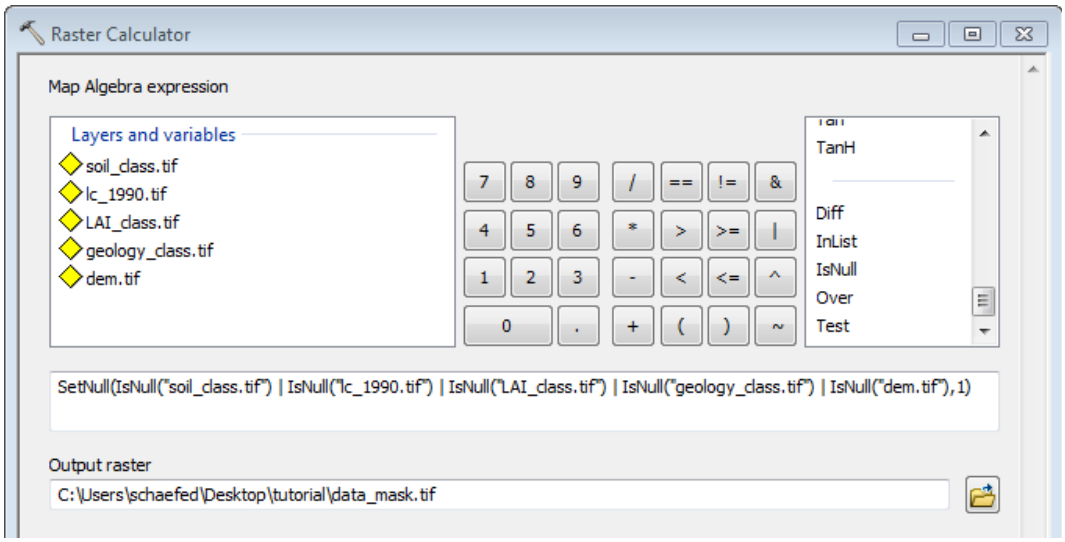


Figure 3.5: System Toolboxes -> Spatial Analyst Tools -> Map Algebra -> Raster Calculator

- Mask all the mentioned datasets with the output of the 'Raster Calculator' follwing the procedure described in Mask the datasets of this tutorial. In case the described processing step is necessary, accomplish it **before** you reach subsection Flow direction and flow accumulation ! Masking these maps would most likely disturb the hydrological properties of your catchment data and result in unexpected model behaviour.

3.5.2 Slope map

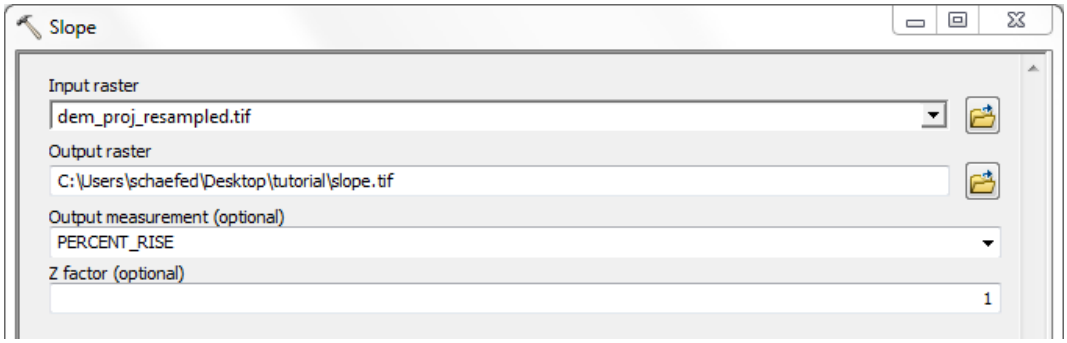


Figure 3.6: System Toolboxes -> Spatial Analyst Tools -> Surface -> Slope

3.5.3 Aspect map

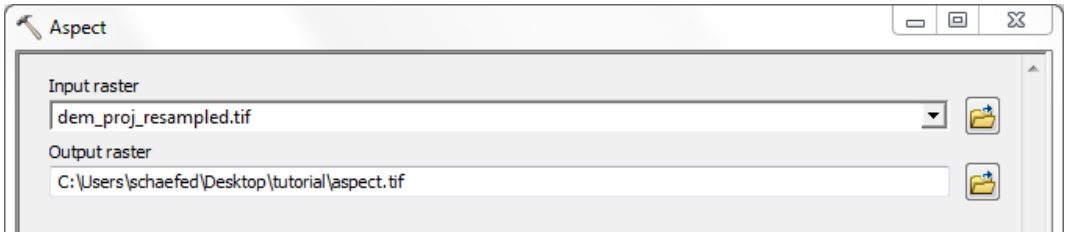


Figure 3.7: System Toolboxes -> Spatial Analyst Tools -> Surface -> Aspect

3.5.4 Fill DEM sinks

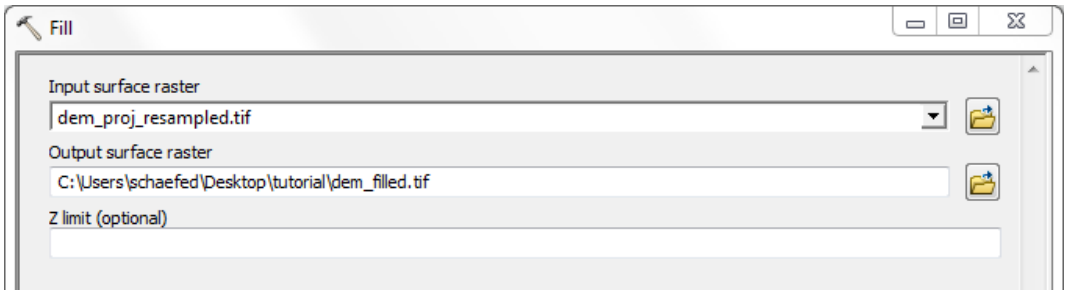


Figure 3.8: System Toolboxes -> Spatial Analyst Tools -> Hydrology -> Fill

3.5.5 Flow direction and flow accumulation

Depending on quality and resolution of the DEM map, these steps can be done with the respective tools from the Spatial Anaylst Extension or by using the Arc Hydro Tools.

3.5.5.1 Spatial Analyst

If a high quality DEM, with a resolution fine enough to represent small scale river morphology is available, you may calculate flow direction and flow accumulation directly.

- Flow Directon

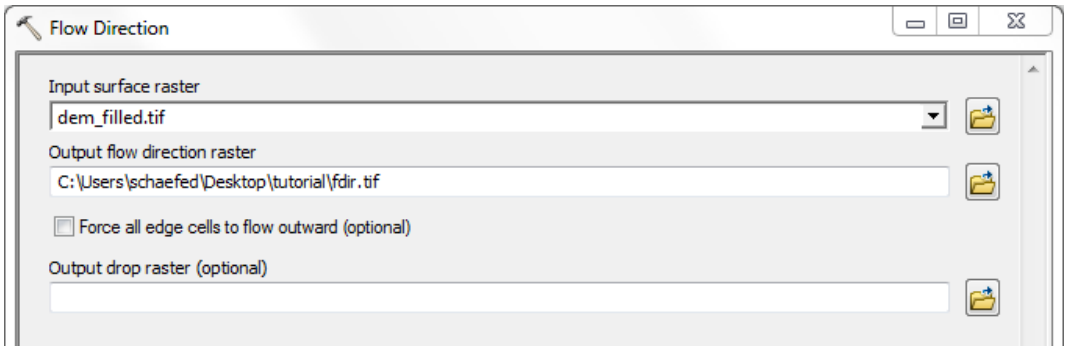


Figure 3.9: System Toolboxes -> Spatial Analyst Tools -> Hydrology -> Flow Direction

- Flow Accumulation

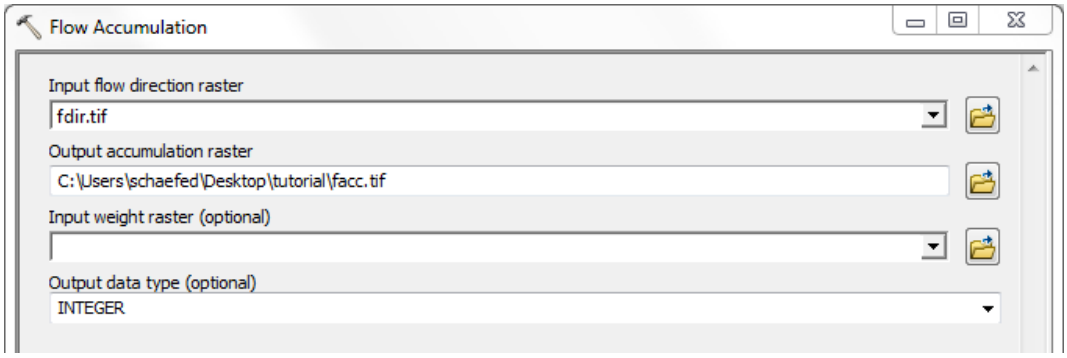


Figure 3.10: System Toolboxes -> Spatial Analyst Tools -> Hydrology -> Flow Accumulation

3.5.5.2 Arc Hydro Tools

Using the Arc Hydro Tools is recommended in case of doubtful DEM quality and/or coarse map resolutions.

- In a first step the original DEM must be reconditioned, i.e. the given altitudes will be reassigned in dependence of a stream network. The latter must be given as a Line Shapefile. In case the necessary stream network file is not available, you can get one from the USGS HydroSHEDS download portal http://hydrosheds.cr.usgs.gov/dataavail.php.

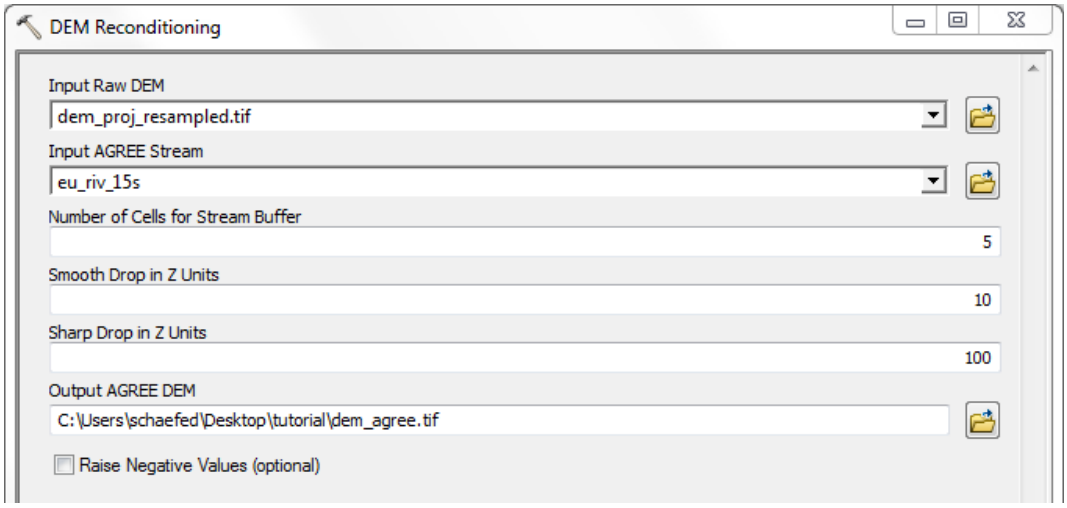


Figure 3.11: System Toolboxes -> Arc Hydro Tools -> Terrain Preprocessing -> DEM Reconditioning

- Fill the sinks in the resulting reconditioned DEM

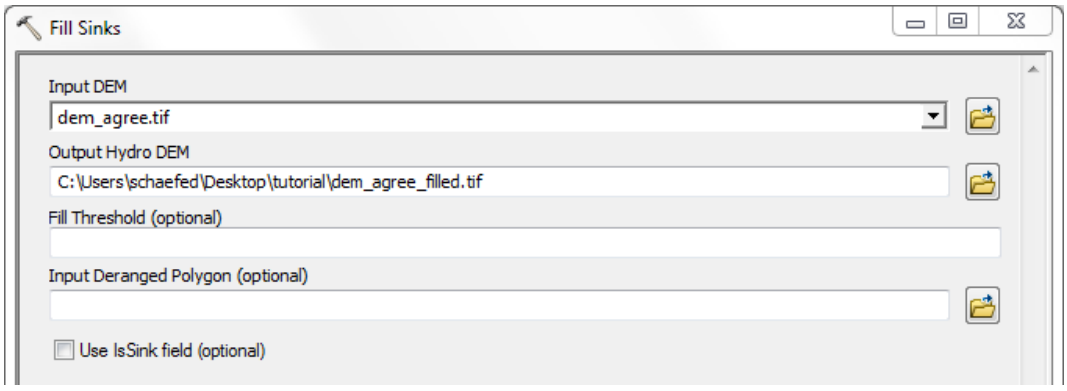


Figure 3.12: System Toolboxes -> Arc Hydro Tools -> Terrain Preprocessing -> Fill Sinks

- Calculate Flow Direction

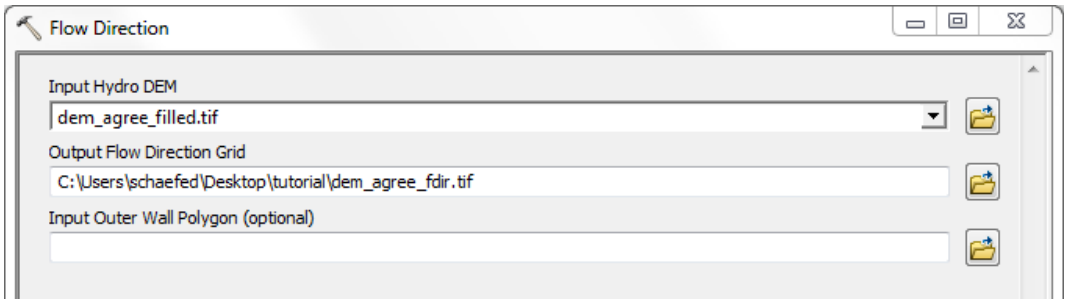


Figure 3.13: System Toolboxes -> Arc Hydro Tools -> Terrain Preprocessing -> Flow Direction

- Create a Flow Accumulation map

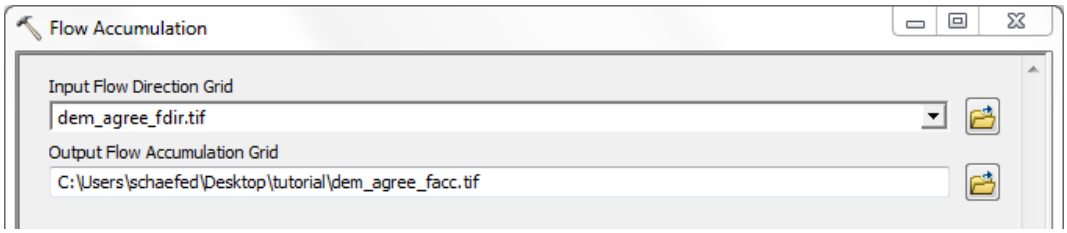


Figure 3.14: System Toolboxes -> Arc Hydro Tools -> Terrain Preprocessing -> Flow Accumulation

3.5.6 Gauges map

- Assuming that you have the positions of your gauges in a table, which has at least the colums x, y, id (in any order and/or amongst other columns), you are able to to convert your data into a Point Shapefile. Choose the appropiate fields and do not forget to set the coordinate system information.

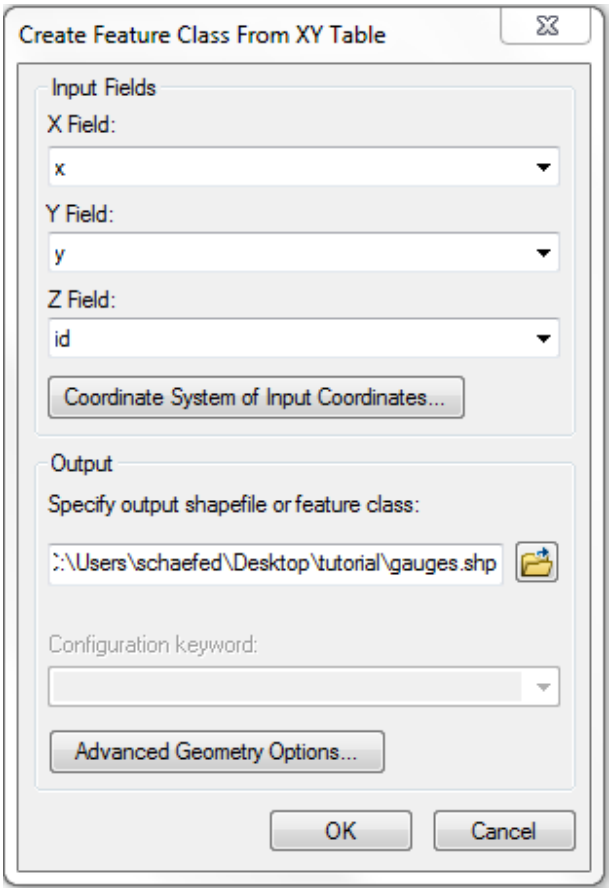


Figure 3.15: Right click on the table in Arc Catalog - > Create Feature Class -> From XY Table

- Check the positions of your gauges against the previously created Flow Accumulation map. If the points do not exactly match the stream-like features on the Flow Accumulation grid, edit the Shapefile (right click on the Point Feature in the Table of Contents -> Edit Features -> Start Editing) and move the gauges to do so. Changing the depiction of the Flow Accumulation map might facilitate this step (i.e. right click the Flow Accumulation map in the Table of Contents -> Properties -> Tab 'Symbology' -> Choose 'Stretched' in the 'Show' box on the left hand side and Type 'Standard Deviations' in the center box. It might also be necessary to invert the color ramp by (un-)checking the 'Invert' check box). Remember to stop the editing process and save your edits (Editor Toolbar -> Editor -> Stop Editing).

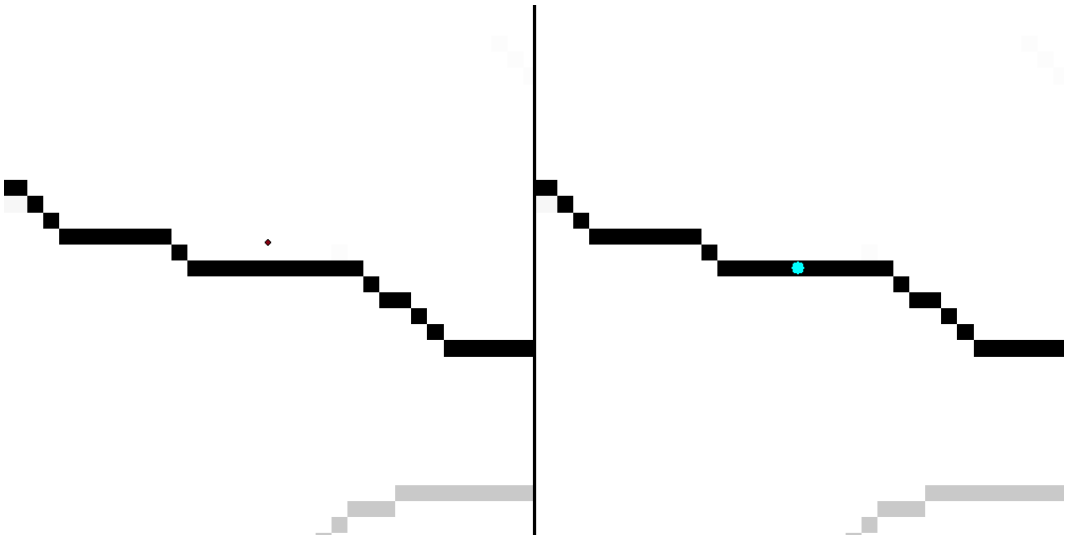


Figure 3.16: Mismatching gauges (left) should be corrected (right)

- Convert the resulting shapefile into an raster map with level-0 resolution.

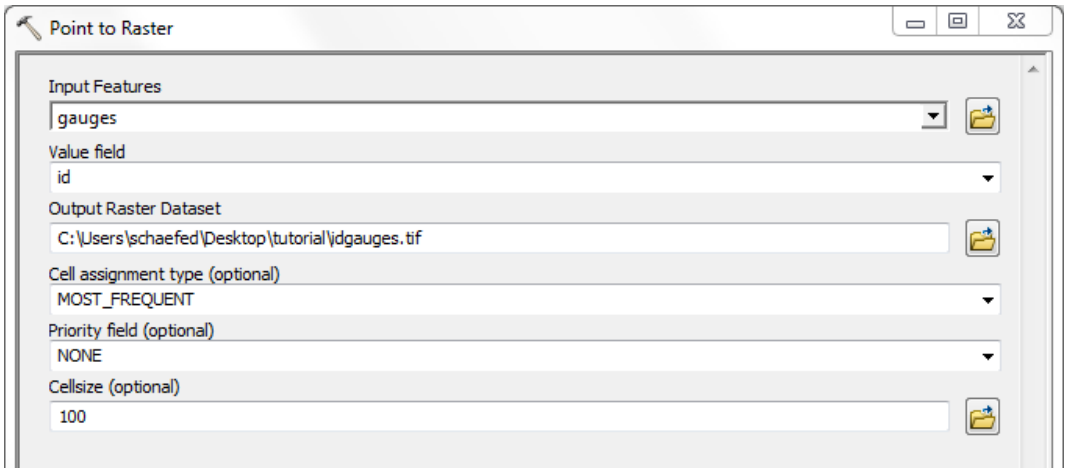


Figure 3.17: System Toolboxes -> Conversion Tools -> To Raster -> Point to Raster

3.5.7 Watershed delineation

- Edit the (possibly corrected) gauges shapefile created during the last processing step again and remove all but the outlet gauge. If you need the unedited file, create a copy of the original Shapefile before editing it.

- Delineate the basin

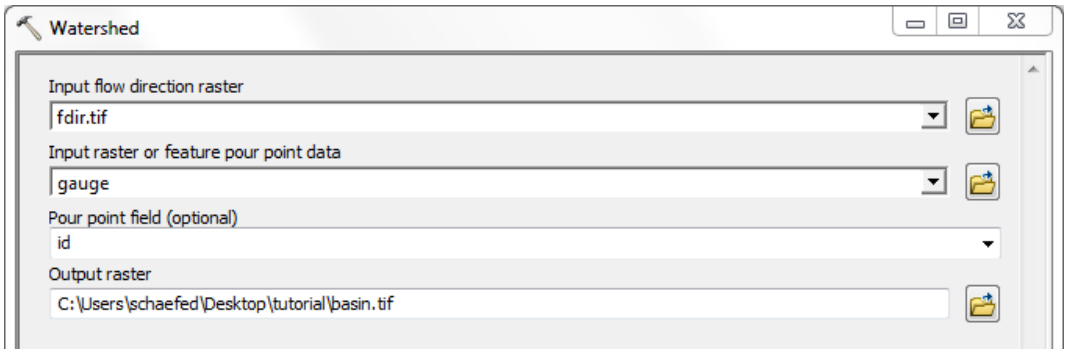


Figure 3.18: System Toolboxes -> Spatial Analyst -> Hydrology -> Watershed

3.5.8 Mask the datasets

Mask all mHM input raster files with the created watershed mask

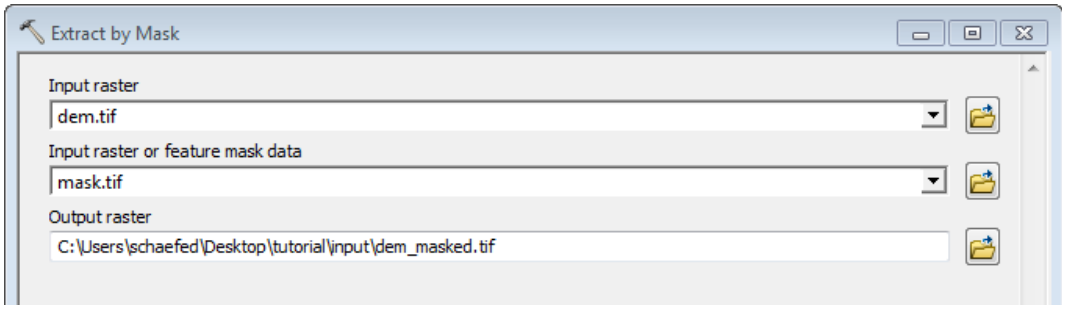


Figure 3.19: System Toolboxes -> Spatial Analysis Tools > Extraction > Extract by Mask

3.5.9 Write the ascii grids

Convert the processed and masked raster maps into ASCII files

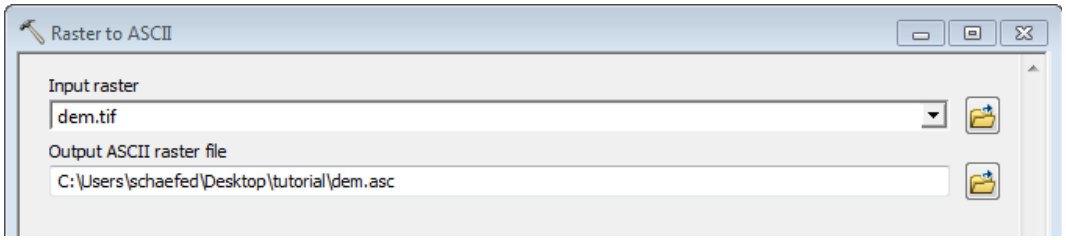


Figure 3.20: System Toolboxes -> Conversion Tools -> From Raster -> Raster to ASCII

3.6 Land Cover Data

Unlike the other classified datasets (soils, hydrogeology, LAI), where as much information as available can be added, the land use data is restricted to three different classes, which are listed below:

Class	Description
1	Forest

Class	Description
2	Impervious
3	Pervious

3.7 Table Data

As previously stated, the input datasets 'soil_class.asc', 'geology_class.asc', 'idgauges.asc' and 'LAI_class.asc' need to be complemented by look-up-tables. All these look-up tables specify the total number of classes/units to read in the very first line, following the keywords 'nSoil_Types', 'nGeo_Formations' and 'NoLAIclasses' respectively. The second line acts as a header describing the contents of the following data. In the subsection The soil look-up table hydrogeo_table and The LAI look-up table screenshots of shorted, but sufficient table files are presented. All fields are further listed and described in the repective tables.

3.7.1 The soil look-up table

```
nSoil_Types  2
MU_GLOBAL      HORIZON UD[mm]  LD[mm]  CLAY[%] SAND[%] BD[gcm-3]
1              1       0       300     23.0    38.0    1.2
1              2       300     1000    31.0    25.0    1.4
2              1       0       50      19.85   53.45   1.41
2              2       50      300     31.33   45.22   1.5
2              3       300     1450    35.42   39.8    1.67
```

Figure 3.21: A possible 'soil_classdefinion.txt' file

Field	Description
MU_GLOBAL	Soil id as given in the corresponding 'soil_class.asc' map. All raster map ids must be given here
HORIZON	Horizon number starting at the top of the soil column. Your are free to provide any number of horizons for each soil individually
UD[mm]	Upper depth of the horizon in millimeter. Should be 0 for the first, and the lower depth of the superimposing horizon for all others
LD[mm]	Lower depth of the horizon in millimeter
CLAY[%]	Clay content in percent
SAND[%]	Sand content in percent
BD[gcm-3]	Mineral bulk density in grams per cubic centimeter

3.7.2 The hydrogeolgy look-up table

```
nGeo_Formations  2
GeoParam(i)   ClassUnit     Karstic      Description
         1            1           0      GeoUnit-1
         2            2           0      GeoUnit-2
```

Figure 3.22: A possible 'geology_classdefinion.txt' file

Field	Description
GeoParam(i)	Parameter number of the formation, the link to 'mhm_parameter.nml'

Field	Description
ClassUnit	Class id as present in the 'hydrogeology_class.asc' raster map
Karstic	Presence of karstic formation (0: False, 1: True)
Description	Text description of the unit

3.7.3 The LAI look-up table

```
NoLAIclasses          10
ID   LAND-USE              Jan.   Feb.   Mar.   Apr.   May    Jun.   Jul.   Aug.   Sep.   Oct.   Nov.   Dec.
 1    Coniferous-forest    11     11     11     11     11     11     11     11     11     11     11     11
 2    Deciduous-forest     0.5    0.5    1.5    4.0    7.0    11     12     12     11     8.0    1.5    0.5
 3    Mixed-forest         3.0    3.0    4.0    6.0    8.0    11     11.5   11.5   11     9.0    4.0    3.0
 4    Sparse-forest        2.0    2.0    3.0    5.5    6.5    7.5    7.5    7.5    6.5    4.0    2.5    2.0
 5    Sealed-Water-bodies  0.01   0.01   0.02   0.04   0.06   0.09   0.09   0.07   0.06   0.04   0.02   0.02
 6    Viniculture          1.0    1.0    1.0    1.5    2.0    3.5    4.0    4.0    4.0    1.5    1.0    1.0
 7    Intensive-orchards   2.0    2.0    2.0    2.0    3.0    3.5    4.0    4.0    4.0    2.5    2.0    2.0
 8    Pasture              2.0    2.0    3.0    4.0    5.0    5.0    5.0    5.0    5.0    3.0    2.5    2.0
 9    Fields               0.4    0.4    0.3    0.7    3.0    5.2    4.6    3.1    1.3    0.2    0.0    0.0
10    Wetlands             2.0    2.0    3.0    4.0    5.0    5.0    5.0    5.0    5.0    3.0    2.5    2.0
```

Figure 3.23: A possible 'LAI_classdefinion.txt' file

Field	Description
ID	LAI class id as given in the file LAI_class.asc
LAND-USE	Description of the LAI class
Jan. to Dec.	Monthly LAI values

3.7.4 The gauge files

```
00139:BRUGG/AARE (Abfluss)     DAILY
nodata  -9999
n       1       measurements per day [1, 1440]
start  1996  01  01  00  00   (YYYY MM DD HH MM)
end    1996  01  06  00  00   (YYYY MM DD HH MM)
1996  01  01  00  00    368.500
1996  01  02  00  00    408.100
1996  01  03  00  00    493.600
1996  01  04  00  00    502.500
1996  01  05  00  00    453.100
1996  01  06  00  00    439.600
```

Figure 3.24: A possible gauge file

The structure of the gauge files is different from the look-up tables listed so far. The following table describes its content.

Line	Description
1	Gives basic information about the gauging station (Station-Id:Station-Name/River-Name)
2	Specifies the nodata value
3	The number of measurements per day
4	The start date of the time series in the format given in parentheses
5	The end date of the time series in the format given in parentheses

Line	Description
6 to end	The measurement data in cubic meter per second preceded by the actual date in the same format as the dates of lines four and five

For every gauge id given in 'idgauges.asc' one gauge file has to be created as outlined above. The file name has to **exactly** reflect the gauge id and carry a '.txt' extension. The table data file for a gauge with an id of 0343 in 'idgauges.asc' should therfore be named '0343.txt'.

3.8 Post-GIS preparation

Make sure that all decimals are indicated with dots, not commas:

```
perl -i.bak -pe 's/\,/\./g' *.asc
```

Make sure that accuracy of your llcorners is reasonable. For example, the following code removes all but one digits after the dot:

```
perl -i.bak -pe  's/(^[xy].+)\.(\d)\d+/$1\.$2/g' *.asc
```

Make sure that your files cover the same spatial domain. In case your grid origins match, but you are missing rows and columns, the perl script `pre-proc/enlargegrid.pl` will pad your data at the top and the right end of the grid.

```
/basin/input/morph/$ ~/mhm/pre-proc/enlargegrid.pl aspect.pl
 31 rows with 47 cols.
/basin/input/morph/$ ~/mhm/pre-proc/enlargegrid.pl 50 40
 aspect.asc
 dem.asc
 facc.asc
 fdir.asc
/basin/input/morph/$ ~/mhm/pre-proc/enlargegrid.pl aspect.pl
 40 rows with 50 cols.
```

If your data differs more substantially the program 'mHM/pre-proc/resize_grid.py' will harmonize your grids.

```
python match_grids.py source_grid target_grid out_grid
```

The script takes three Input arguments:

1. The source grid to be enlarged
2. The target grid which could also be an header file as described in subsection The Header File
3. An output file

The program will fail, if your target grid is smaller than the source grid or if the cellsizes of both grids are not divisable. If necessary the source grid will be shifted in order to make both origins match. This 'shift' is only accomplished by changing the values of the corner coordinates, no real interpolation will be done.

Chapter 4

Visualizing Model Output

mHM usually writes two files into the output directory specified in mhm.nml:

```
ConfigFile.log
daily_discharge.out
dischage.nc
mHM_Fluxes_States.nc
mRM_Fluxes_States.nc
```

In addition to these, three restart files are saved into the restart directory also specified in mhm.nml:

```
xxx_config.out
xxx_L11_config.nc
001_states.nc
```

The first file ConfigFile.log is ASCII type and summarizes the main configuration of mHM. The second file daily_discharge.out is ASCII type and can be read by any standard editor or ASCII interpreting scripts. It contains the simulated and observed discharge. The same timeseries are stored for all gauges in discharge.nc (the third file), but in NETCDF format. In mHM_Fluxes_States.nc all the spatial and temporal output is written, with the exception of routed discharge which can be found in mRM_Fluxes_States.nc (also a NETCDF file). These binary NETCDF files can be visualised with NCVIEW (part of the NETCDF library), VISIT (LLNL Software) or analysis scripts based on Python (PyNGL). Please note that daily_discharge.out, discharge.nc and mRM_Fluxes_States will only be written if the routing is switched on (i.e., process_case 8 equals one).

The restarting files are mainly intended for improving model performance (i.e,. to avoid spin-up time) rather than for visualization purposes. The xxx prefix indicate the basin number. Since these files are also written into a binary NETCDF file, some remarks will be make below for the right interpretation of their content.

For a quick analysis we recommend NCVIEW since it quickly visualises those files via command-line and X-forwarding. Make sure to load the NC module before starting.

```
ncview FILE.nc
```

In order to hide unnecessary output messages, you may pipe them to a log file:

```
ncview FILE.nc 1> ~/log/ncview.out 2> ~/log/ncview.err
```

If you want to transfer large nc files through servers or visualise them locally, it might be useful to compress the data before. The featured nc file deflation with the `ncks` module will decrease the file size usually by 30-60%. Choose the deflation level from minimum (0) to maximum (9). The `-4` switch in the following command will convert netcdf3 files to version 4 simultanously.

```
ncks -4 -L 9 IN.nc OUT.nc
```

Using X-Forwarding under Windows

On Windows we recommend CYGWIN including MINTTY, OPENSSH and XINIT, XMING as the X server. An alternative is PUTTY using X-forwarding. The workflow in MINTTY is as follows:

```
/bin/XWin.exe -multiwindow
ssh -Y SERVERNAME
module load ncview
ncview FILE.nc
```

In some cases it has proven to be necessary to add the following line to /etc/profile

```
export DISPLAY=:0
```

Exploring the contents of the Restarting files

Restarting files are simple binary dumps of arrays and vectors of all constants, parameters, state variables (1D, 2D) and fluxes at a given point in time of a simulation that are needed for executing the subsequent mHM time step in a new instance of this model without performing spin-out simulations and additional run time up to the previos time point. The stored information is divided into three categories: 1) Configuration variables at L0 and L1 levels (xxx_config.nc), 2) configuration variables at L11 level (i.e., routing) (xxx_L11_config.nc), and 3) and effective parameters, state variables and fluxes at L1 and L11 levels (xxx_states.nc).

WARNINGS

1) Results may appear inverted in NCVIEW with respect to the geographical coordinates used to describe the basin domain. This is due to the lack of geographical coordinates in this NetCDF visualizer and the fact that it simply dumps the content of a 2D array in C convection (row first, DIM2). In addition to that, mHM 2D arrays are stored in (lon, lat) or (X,Y) ordering, which is the transpose of the ESRI convention (lat, lon) or (Y,X).

2) Caution should be taken to interpret flow direction variables (i.e., variable L11_fDir in xxx_L11_config.nc) if visualized with NCVIEW because of its implicit 90 $^\circ$ counter-clockwise rotation. Hence the values depicted NCVI↩ EW using 8-flow direction convention (1,2,4,8,16,32,64,128) (i.e., 1 pointing to the east and then moving clockwise every 45 $^\circ$) have to be rotated accordingly. In other words, direction 1 should be interpreted as 64, 2 as 128, and so on. All 2D variables, however, included the previous one, will produce consistent maps as that shown in (Moselle Basin) if plotted with appropriate routines (e.g., Python or NCL) that take into account the geographic coordinates stored within the NetCDF file.

Chapter 5

Calibration Options

5.1 Calibration of Parameters on Observations

In order to use the calibration feature of mHM, turn the `optimize` switch in `mhm.nml` to `.true.`.

5.1.1 Parameters

MHM calibrates all parameters in `mhm_parameters.nml` which are flagged for optimization (column "FLAG"). Parameters can be forced to stay constant during calibration by setting the flag to 0. The upper and lower bounds define the ranges in within which the optimization methods vary the parameters. Although permitted, we recommend not to change the bounds, since they are the result of intensive sensitivity studies covering a wide range of basins.

5.1.2 Methods

Different optimization methods are available to find the best configuration of parameters for the selected objective function. You may chose between Simulated Annealing (SA), Dynamically Dimensioned Search (DDS) and Shuffled Complex Evolution algorith (SCE). Details about these methods can be found in the module describtion part of this manual. At the very end of `mhm.nml` additional settings are offered for optimization.

5.1.3 Functions

The measure to define how well a quantity fits to the observations depends strongly on the specific type of application. mHM offers a variety of options for different quantities like discharge, soil moisture, or total water storage. For example, an NSE type of function will not be able to catch low values as well as ln(NSE) would. In `mhm.nml`, individual objective functions are defined for specific quantities.

5.1.4 Data

Discharge and total water storage data should be provided in ASCII file format for each gauge/basin. Soil moisture and neutron data need to be provided as spatio-temporal netcdf files. Example files can be found in thr folder `optional_data` in the `test_basin`. Furthermore, a `optional_data` namelist in `mhm.nml` provides several options regarding the handling of such data. For example, for soil moisture data, the temporal resolution needs to be set in `mhm.nml`. However, neutron data is restricted to be daily in the current release. The spatial resolution of soil moisture and neutrons data needs to match the hydrological resolution of mHM.

For correct preparation of input data, please take a look at the input data in the `test_basin`, which will serve as a good example. Furthermore, it could be helpful to generate optional data files from precedent mHM output, because then the temporal resolution and the spatial grid is guaranteed to match the needs of mHM. The files then can be modified with tools like `cdo` or `python`.

5.1.5 Output

Calibration runs result in a parameter file called `FinalParams.nml` which contains the optimized parameter set. Replace `mhm_parameters.nml` with `FinalParams.nml`, then run mHM in foreward mode in order to obtain hydrologic predictions.

Chapter 6

Coding and Documentation Style

The coding and documentation style guide has the following sections:
General
In and Out of Files
Modules, Subroutines and Functions
Variables
Documentation

General

- **Follow the coding and documentation style of template mo_template.f90 very closely.**
- Enforce SI units in all code, with the only exception of $\mathrm{mm s^{-1}}$ for $\mathrm{kg\ m^{-2} s^{-1}}$.
- mHM is computed in double precision (dp).

```
real(dp) :: wind_speed ! wind speed in [m s-1]
```

- New routines should be tested with at least two different compilers (of different vendors).

In and Out of Files

- Filenames are all lower case.
- Module names and filenames are the same and start with mo_
- Filenames should be descriptive of the content and use as little abbreviations as possible.
- Fortran filenames end with .f90

```
mo_string_utils.f90
```

- Never use tabs.
- Break lines at column 130, at most. This includes comments

```
sum_of_all = a + b + c + d + e + f + g + h + i + j + k + l + m + n + o + p + q + r + s + &
             t + u + v + w + x + y + z
zero       = 0.0_dp ! This literal can be used whenever something has to be initialised, so that
                    ! it will be initialised with the right number precision
```

- Indent all code blocks.

```
if (abs(a-b)>epsilon(1.0_dp)) then
  if (a >= b) then
    a = b
  else
    b = a
  endif
endif
```

Modules, Subroutines and Functions

- All modules and programs have IMPLICIT NONE.
- All USE statements have the only clause.

```
module mo_calc

  use mo_kind, only: i4, dp

  implicit none

  ...
```

- Modules are PRIVATE by default.
- Module routines are made available explicitly, i.e. PUBLIC.
- Give 1-line descriptions after the public definition.

```
module mo_calc

  use mo_kind, only: i4, dp

  implicit none

  private

  public :: calc ! Calculates the thing

contains

  function calc(x)

  ...
```

- Try to write elemental pure subroutines whenever possible (see below).
- Extremely short subroutines should be avoided.
- Avoid very long subroutines ($>$500 lines).

Variables

- Use expressive and descriptive variable names for all variables in the namelist and for all variables that have a larger scope, i.e. that are used in more than a screen full.
  This means that counter variables can still be i, ii, j, ... and local variables can also be called short names such as tmp or itmp.
  But variables, which are important to read the formula, should have descriptive names such as ido_that or iDoThat.
  Variable names should not be too long either. The agreed maximu numbers are
  - filenames $<$ 30 characters
  - variable names $<$ 20 characters
- All variables and literals use explicit type declarations from mo_kind.

```
elemental pure function circum(radius)

  use mo_kind,     only: dp
  use mo_constans, only: pi_dp

  implicit none

  real(dp), intent(in) :: radius
  real(dp)             :: circum

  circum = 2.0_dp * pi_dp * radius

end function circum
```

- Array dimensions are in the following order: lon (east-west), lat (north-south), z (vertical), t (time level). Additional dimensions can be used and are placed in front of t.
- Always pass arrays as sub-program arguments. This means: use as little global variables as possible.
- All input/output arguments have an intent.

```
subroutine calc(x,y)

  use mo_kind, only: i4, dp

  implicit none

  real(dp),    dimension(:,:),                intent(in)  :: x
  integer(i4), dimension(size(x,1),size(x,2)), intent(out) :: y

  ...
```

- Use intuitive names for optional arguments.

```
function calc(x, inverse)

  use mo_kind, only: i4, dp

  implicit none

  real(dp),    dimension(:,:),     intent(in) :: x
  logical,     optional,           intent(in) :: inverse
  integer(i4), dimension(size(x,1),size(x,2)) :: calc

  logical :: iinverse

  iinverse = .false.
  if (present(inverse)) then
    iinverse = .false.
    if (inverse) iinverse = .true.
  endif

  ...
```

- Global variables are initialised at the setup stage.
- Never use hardcoded number literals such as 5., 3.14, etc. Exceptions might be -1, 0, 1, and 2. But all literals are appended by _dp, _i4, etc.

```
elemental pure function circum(radius)

  use mo_kind,     only: dp
  use mo_constans, only: pi_dp

  implicit none

  real(dp), intent(in) :: radius
  real(dp)             :: circum

  circum = 2.0_dp * pi_dp * radius

end function circum
```

- Avoid pointers whenever possible.

- Index notation is encouraged whenever whole arrays are treated such as a(:) = b(:)∗c(:).
  It is still very good for contiguous subarrays such as a(1:n-1) = b(1:n-1)∗c(1:n-1).
  It is discouraged, though, for mixed indexing such as a(1:n-1) = b(2:n)∗c(1:n-1). This is (1) prone to coding error, (2) not easy to read by other programmers, and (3) hard to parallelise with OpenMP or MPI. Use forall or do instead.

Documentation

Follow the documentation structure before the interface mean in mo_template.f90.
Documentation uses the automatic documentation system doxygen (http://www.doxygen.org/). See the manual for the complete list of documentation tags: http://www.doxygen.nl/manual.html

- The documentation of a routine is in front of the routine definition.
- Documentation for a module interface must be in front of the interface definition.
  The individual routines do not need extra documentation.
- Only public elements will be considered by the automatic documentation system doxygen.
- Make your routines public at the beginning of the file. Append a one-line descriptions to the public statement.

Chapter 7

The details about the test basin

The test data coming with the mHM source code are for the Moselle River basin upstream of Perl, a place, where the Moselle River leaves France and enters Luxembourg and Germany (Moselle Basin). The catchment area is approximately 11,500 km ^{2}, altitude ranges between 150 and 1300 m. a.m.s.l. The Moselle River originates from the Vosges Mountains and is a tributary of the Rhine River. The origin of data used in the test example is listed below.

Meteorological forcings (temperature and precipitation):
We acknowledge the E-OBS dataset from the EU-FP6 project ENSEMBLES (http://ensembles-eu.↩ metoffice.com) and the data providers in the ECA&D project (http://www.ecad.eu).
"Haylock, M.R., N. Hofstra, A.M.G. Klein Tank, E.J. Klok, P.D. Jones, M. New. 2008: A European daily high-resolution gridded dataset of surface temperature and precipitation. J. Geophys. Res (Atmospheres), 113, D20119, doi:10.1029/2008JD10201".
(http://www.ecad.eu/download/ensembles/ensembles.php 02.04.2014)

Hydrological observations:
We acknowledge the Global Runoff Data Centre (GRDC), a repository for the world's river discharge data and associated metadata for providing the discharge data.
(http://www.bafg.de/GRDC/EN/01_GRDC/12_plcy/policy_guidelines.pdf;jsessionid=↩ AA96F6F200B3880575879F15534B6669.live2052?__blob=publicationFile 02.04.2014)

SRTM (Shuttle Radar Topography Mission):
We acknowledge the U.S. Geological Survey's Earth Resources Observation and Science (EROS) Center or NA↩ SA's Land Processes Distributed Active Archive Center (LP DAAC) for providing the digital elevation model.
(http://www2.jpl.nasa.gov/srtm/cbanddataproducts.html 15.05.2014)

Harmonized world soil database:
We acknowledge the use of Harmonized World Soil Database dataset of the Food and Agriculture Organization of the United Nations (FAO) the International Institute for Applied Systems Analysis (IIASA), International Soil Reference and Information Centre (ISRIC), Institute of Soil Science – Chinese Academy of Sciences (ISSCAS) and Joint Research Centre of the European Commission (JRC).
(http://webarchive.iiasa.ac.at/Research/LUC/External-World-soil-database/HT↩ ML/ 02.04.2014)

European soil database:
We acknowledge the European Commission for providing the European soil database.
(http://ec.europa.eu/geninfo/legal_notices_en.htm 02.04.2013)

Land cover:
We acknowledge the European Environment Agency for providing the CORINE land cover dataset.

(http://www.eea.europa.eu/publications/COR0-landcover – 15.04.2014)

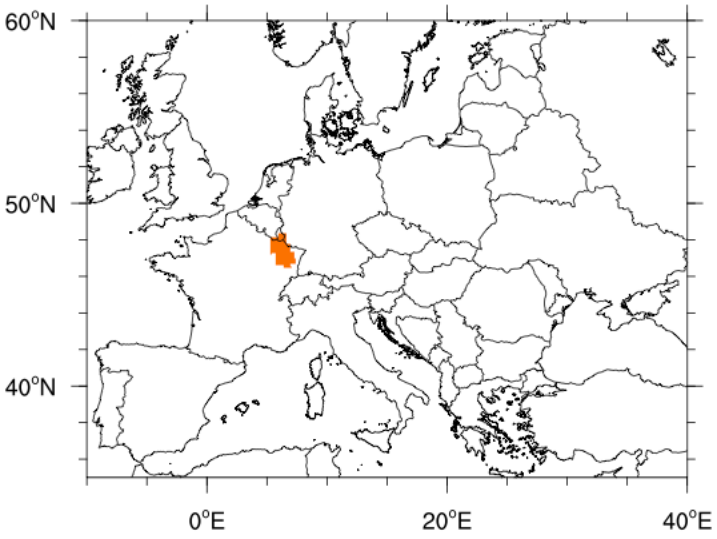


Figure 7.1: Location of the Moselle River basin upstream of Perl within the European domain

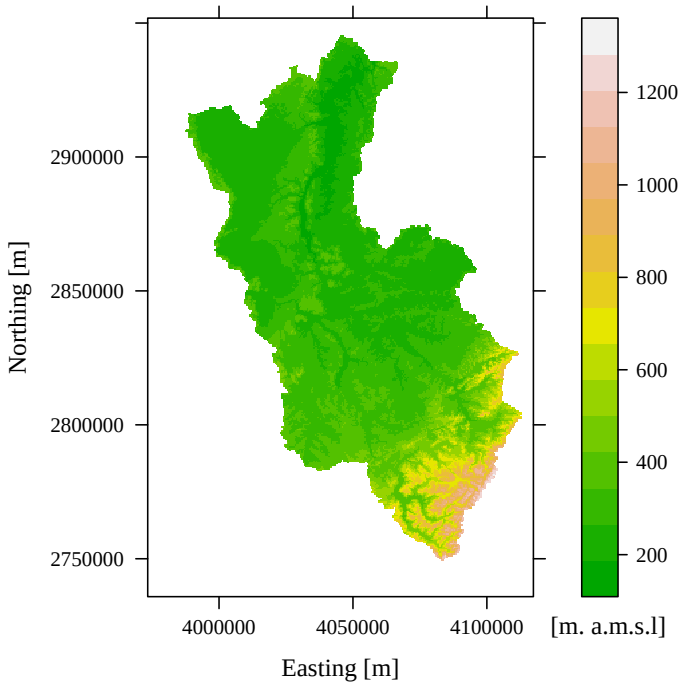


Figure 7.2: Digital elevation model for the Moselle River basin upstream of Perl

Chapter 8

Protocols for setting up a new mHM basin

The following checks and protocols describe some standard procedures for the setup of a new basin in mHM. They are based on the personal experience of the authors and do not cover all possible subjects. They are to be seen as a first guideline when working with a new basin in mHM but do not represent a complete instruction.
This chapter is divided into the following sections:

- check for input data errors
- protocol for generating a restart file
- protocol for determining the warm-up period for a new basin

8.1 Check for possible input data errors using mHM outputs

Besides mHM check for the possible input data errors (in the mo_startup routine), there could still be room for the data errors. Before you make the exhaustive model runs for calibration, make sure that you have processed your input data correctly. Below are some tips for detecting data problems using the mHM outputs.

1. Make a default mHM simulation for a relatively longer time-period, as supported by the input data sets (say 20-30 years).
2. Print out distributed fluxes/states, for example, at a monthly time scale. This feature is supported in the current mHM version using the name list file "mhm_outputs.nml". You just need to tick ".TRUE." to the fluxes/ states you want to save as a netcdf file.

   Check at-least the following model outputs:

   (a) height of snow pack (L1_snowpack) [mm] – case 2 outputFlxState(2)=.TRUE.
   - This variable should be check in those basins which have snow covers/glaciers.
   - Snowpacks in the mountainous/glacier regions should be higher than those of other areas.
   - Snowpack during winter should be higher than that during summer.

   (b) mean volumetric soil moisture averaged over all soil layers [mm/mm] – case 5 (= L1_soilMoist / L1_soilMoistSat) outputFlxState(5)=.TRUE.
   - Soil moist. during winter should be higher than that of during summer.
   - In general, soil moist. in hilly areas is higher compared to that in flatter areas

   (c) actual evapotranspiration aET [mm/T] – case 9 outputFlxState(9)=.TRUE.
   - Evapotrans. during winter should be lower than that of during summer.
   - In general, evapotrans. in hilly areas is lower compared to that in flatter areas

   (d) total discharge generated per cell (L1_total_runoff) [mm/T] – case 10 outputFlxState(10)=.TRUE.

- In general, runoff in hilly areas is higher compared to that in flatter areas

3. Check the temporal dynamics of modeled fluxes/states across several locations (i.e., at several grid cells).
   - They should, in general, exhibit distinct behavior in the temporal dynamics.
4. You could calculate a long-term mean annual dynamics of variables like
   (a) actual evapotranspiration
   (b) total runoff
   (c) precipitation (use values provided in the meteo-input file)
   - This could fairly give you an idea about the respective water balance regime of the modeled basin. For example, hilly areas should exhibit a higher runoff, as a result of a relatively higher precipitation and lower ET values, compared to those of the flat areas.
   - Compute the values of runoff coefficient (long-term mean annual runoff / long-term mean annual precipitation) These values should be lesser than 1.0 everywhere.
   - Compute the values of long-term mean annual evapotranspiration / long-term mean annual pot. evapotranspiration These values should be also lesser than 1.0 everywhere.
5. Compare the respective spatial patterns of each variable with some known literature results (google them for your respective basins)

Above check list are just few (based on the personal experience of the authors) among many possible ones, and the list will be updated as soon as we find new ways to deal with data problems.

8.2 Protocol for generating a restart file

Not specified yet.

8.3 Protocol for determining the warm-up period for a new basin

Not specified yet.

Chapter 9

mRM - the Multiscale Routing Model: Running the stand-alone version of the routing model

9.1 Introduction to mRM

This section provides information on how to run the routing model mRM as a stand-alone program outside of mHM. The general philosophy is as follows: mRM can be compiled and run in a similar way as mHM. Specific files for mRM can be found in the main path and are denoted as <filename>_mRM.<file_type>, where <filename>.<file_type> is the corresponding mHM file.

This Subsection provides a short description of the method used in mRM. For further details and an analysis over the European domain, please refer to Thober et al. 2019.

9.1.1 Finite Difference Approximation of Kinematic Wave Equation

The multiscale Routing Model mRM uses the kinematic wave equation to describe the water flow within a stream as

$$\frac{\partial Q}{\partial t} + c\frac{\partial Q}{\partial x} = 0 \tag{9.1}$$

where Q (m ^{3}\,s $^{-1}$) is streamflow, x (m) the space dimension, t (s) the time dimension, and c (m\,s $^{-1}$) the celerity. The kinematic wave equation is a simplification of the Saint-Venant equations. The derivation is based on the assumption that the continuity equation is sufficient to describe the movement of flood waves. In detail, a constant river bed slope and time-invariant celerity c have to be assumed. mRM employs the classical finite difference weighted approximation on a four points scheme to solve the equation above. A short derivation is as follows:

The partial derivatives within the above equation are represented as finite differences between four values, that means on two locations at two points in time:

$$\begin{aligned}
\frac{\partial Q}{\partial t} &\approx \frac{\varepsilon\big(Q(x_j,t_{i+1}) - Q(x_j,t_i)\big) + (1-\varepsilon)\big(Q(x_{j+1},t_{i+1}) - Q(x_{j+1},t_i)\big)}{\Delta t}, \\
\frac{\partial Q}{\partial x} &\approx \frac{\varphi\big(Q(x_{j+1},t_{i+1}) - Q(x_j,t_{i+1})\big) + (1-\varphi)\big(Q(x_{j+1},t_i) - Q(x_j,t_i)\big)}{\Delta x},
\end{aligned}$$

where j denotes the spatial location (i.e., reach id) and i denotes the timestep. ε is a space-weighting factor and φ is a time-weighting factor. mRM uses a rectangular grid to represent the river network with a river reach in the model connecting two center grid locations. Different spatial locations are separated by Δx and time steps by Δt. The two weighting factors, ε and φ, can be chosen between 0 and 1, but the numerical solution becomes unstable for $\varepsilon >$ 0.5. The numerical diffusion depends linearly on ε, with 0 implying full numerical diffusion and 0.5 no numerical diffusion, respectively.

Setting φ to 0.5, which represents a time-centered scheme, and substituting the above equation into equation (9.1) results in the classical linear equation:

$$Q(x_{j+1},t_{i+1}) = C_1 Q(x_j,t_{i+1}) + C_2 Q(x_j,t_i) + C_3 Q(x_{j+1},t_i), \tag{9.2}$$

with the coefficients C_1, C_2 and C_3 being:

$$\begin{aligned} C_1 &= \frac{-2\Delta x\varepsilon + c\Delta t}{2\Delta x(1-\varepsilon)+c\Delta t}, \\ C_2 &= \frac{2\Delta x\varepsilon + c\Delta t}{2\Delta x(1-\varepsilon)+c\Delta t}, \\ C_3 &= \frac{2\Delta x(1-\varepsilon) - c\Delta t}{2\Delta x(1-\varepsilon)+c\Delta t}. \end{aligned} \tag{9.3}$$

The coefficients C_1, C_2 and C_3 add up to one. The spatial resolution at which mRM is applied is called “routing" resolution in the following.

9.1.2 Stream Celerity Parametrization based on Terrain Slope

Two parametrizations of equation (9.3) are available in mRM: firstly the regionalized Muskingum-Cunge (rMC) parametrization with a fixed time step, and secondly a parametrization using spatially varying celerities in combination with an adaptive time step (aTS). A short summary of the former is presented in Subsection Regionalized Muskingum-Cunge parametrization and is referred to as rMC in the following. The latter is described in detail in this and the next section and is referred to as aTS scheme.

The aTS calculates stream celerity as a function of terrain slope using a simple relationship:

$$c_i = \gamma\sqrt{s_i}, \tag{9.4}$$

where c_i, s_i and γ are celerity, terrain slope and a free model parameter, respectively, and i is the grid cell index.

The above equation is applied at the resolution of the Digital Elevation Model (DEM), from which terrain slope is derived. A median absolute deviation (MAD) filter is applied to the high resolution slope data along the path of the main river with a threshold value of 2.25 to correct for outliers. Large slope outliers happen easily in DEMs, for example, when the river flows in a valley and one grid cell represents the valley while the next grid cell represents the hill top. A minimum river bed slope of 0.1% is further assumed. The celerities are then upscaled to the routing resolution, i.e., the resolution at which the kinematic wave equation is solved. The upscaling is by averaging with the harmonic mean, the correct averaging operator for celerities (or speed). The upscaling considers also only those high-resolution grid cells that align with the path of the main river because aTS only considers the flow in the main river reach, assuming that travel times in the main reach dominate the routing process in tributaries.

9.1.3 Adaptive Time Step (aTS) Implementation

The aTS scheme uses an adaptive time step to calculate the linear coefficients in equation (9.2). The basic idea is that the time step should be such that water has not been transported further than Δx during a single step. This condition is generally known as Courant-Friedrich-Lewy criterium, which is a necessary condition for numerical stability of finite difference schemes [8]. The condition can be expressed as:

$$C_r = \frac{c\Delta t}{\Delta x} \leq C_{max} = 1 \tag{9.5}$$

where C_{max} is the Courant number. aTS uses a Courant number of 1. The Courant condition couples the applied spatial resolution with the integration time step of the finite difference scheme. Celerities c_i are typically in the order of a few m s $^{-1}$ The time step Δt is chosen such that it does not deviate too much from the Courant number C_{max} to keep computational demand to a minimum. This implies that Δt ranges from a few minutes for high-resolution grids to a few hours for continental scale applications.

In detail, aTS chooses Δt from a set of prescribed values such that $c_i\Delta t/\Delta x$ is close to but less than 1 for all celerities c_i. The prescribed values range from one minute to one day, namely: 1 min, 2 min, 3 min, 4 min, 5 min, 6 min, 10 min, 12 min, 15 min, 20 min, 30 min, 1 h, 2 h, 3 h, 4 h, 6 h, 8 h, 12 h, and 1 day. The choice of these values is

motivated from the fact that these represent multiples or dividers of hourly and daily time steps. These time steps allow in principle model applications from 100 m to 100 km, for typical celerities around 1.5 m s $^{-1}$.

Note that the chosen time step depends only on the spatial resolution and is independent of the time resolution of the provided forcing. For example, applying aTS at 12 km resolution using a celerity of $c\sim=$ 1.5 m s $^{-1}$ gives $\Delta x/c$ of 2.2 hours and, hence, a time step of two hours will be chosen. If mRM is forced with hourly input, it will aggregate the input over two consecutive time steps prior to the routing. The calculated streamflow is then distributed to the previous two time steps because these represent the mean flow over this period. If aTS is forced with daily input, it will use internally 12 iterations of 2-hour time steps to route the water through the river network. In this case, aTS will also return the average of the calculated 12 streamflow values at each reach.

9.1.4 Regionalized Muskingum-Cunge parametrization

The regionalized Muskingum-Cunge (rMC) parametrization implemented in the mesoscale Hydrologic Model mHM calculates the Muskingum coefficients C_1, C_2, and C_3 in equation (9.2) as a function of high-resolution river network properties. The coefficients C_1, C_2, and C_3 are parametrized as follows

$$C_1 = v_2; C_2 = v_1 - v_2; C_3 = 1 - v_1, \tag{9.6}$$

where the parameters v_1 and v_2 are given as

$$\begin{aligned} v_1 &= \frac{\Delta t}{\beta(1-\varepsilon)+\frac{\Delta t}{2}}; \\ v_2 &= \frac{\frac{\Delta t}{2}-\beta\varepsilon}{\beta(1-\varepsilon)+\frac{\Delta t}{2}} \end{aligned} \tag{9.7}$$

following the nomenclature of appendix A2 in [16]. This formulation is identical to equatio (9.3) of the present study, using $\beta = \Delta x/c$ in the equation above and substituting this equation into two equations above. The parameters β and ε are then conceptualized as

$$\begin{aligned} \beta &= \gamma_1 + \gamma_2 L + \gamma_3 S + \gamma_4 C; \\ \varepsilon &= \gamma_5 \frac{S}{max(S)}, \end{aligned} \tag{9.8}$$

where L is the length of the reach, S is the slope of the reach, and C is the fraction of impervious land cover within the floodplains (see table 4 in [12]). Overall, there are five global parameters γ_1 to γ_5 in the above equation that can be chosen by the user. The integration time step is fixed at one hour. To guarantee the numerical stability of the parameterization, the following upper and lower bounds are applied

$$0 < \varepsilon \leq 0.5, \tag{9.9}$$

$$\frac{\Delta t}{2(1-\varepsilon)} < \beta \leq \frac{\Delta t}{2\varepsilon}, \tag{9.10}$$

where Δt is set to one hour.

9.2 Getting Started

For receiving mRM, installing NETCDF, and running mRM under CYGWIN on Windows, please follow Sections Install NETCDF, Run mHM under CYGWIN on Windows, and Compile mHM.

9.2.1 Compiling mRM

The compilation of mRM can be achieved by running the *make* command on the **Makefile_mRM** file. This can be done in two ways:

1. make -f Makefile_mRM
2. renaming of Makefile_mRM to Makefile via

```
mv Makefile_mRM Makefile
```

9.2.2 Test mRM on Example Basin

The received mHM distribution contains an example test basin in the folder

```
test_basin/
```

This folder also contains all necessary data to run mRM. After you compiled mRM, simply run the executable via

```
./mrm
```

Ouput will be written in the

```
test_basin/output_b1
```

9.2.3 Run your own Simulation

As for mHM, there are three mRM configuration files. These are:

- mrm.nml
- mrm_output.nml
- mrm_parameter.nml

Please note that the information contained in these files can also be found in the respective files for mHM because mRM is part of mHM.

9.2.3.1 Main configuration: Path, Periods, Switches

The file `mrm.nml` contains the main configuration for running mRM in your catchments. Since the comments should explain each single setting, this section will only roughly describe the structure of the file. All mandatory namelists have to specified explicitly for both mHM and mRM. Namelists that are specific for mRM are denoted by `_mRM`. Please keep in mind, that paths can be relative, absolute and even symbolic links.

- **project_description:** Defines meta data that is written in the output netcdf files.
- **mainconfig:** Defines number of basins and hydrology resolution.
- **mainconfig_mhm_mrm:** Defines input paths for reading restart files, routing resolution, and optimization.
- **mainconfig_mrm:** Defines file name, variable name, and data convention for runoff input field for mRM.
- **directories_general:** Defines common input and output paths for mHM and mRM.
- **directories_mRM:** Defines paths where gauge information and input runoff fields can be found.
- **processSelection:** Only processCase seven is used for mRM, all other options are ignored.

- **LCover:** defines information on LAI data.
- **time_periods:** Defines simulation periods.
- **evaluation_gauges:** Defines number of gauges per basin and file names containing gauge observations.
- **Optimization:** Defines parameters for optimization.

9.2.4 Output Configuration: Time Steps, States, Fluxes

The file `mrm_outputs.nml` regulates how often (e.g. `timeStep_model_outputs_mrm = 24`) and which variables (fluxes and states) should be written to the final netcdf file `[OUTPUT_DIRECTORY]/mRM_Fluxes_States.nc`. We recommend to only switch on variables that are of actual interest, because the file size will greatly increase with the number of containing variables. During calibration (see Calibration and Optimization) no output file will be written.

9.2.5 Calibration and Optimization

The parameter calibration can be conducted in the same way as for mHM. Please see Section Calibration and Optimization and Section Calibration of Parameters on Observations for details. Parameters contained in the **mrm_parameter.nml** are considered during the optimization.

9.3 Data preparation for mRM

The data preparation for the static data for mRM is identical to that of mHM. It is described in the Chapter "Data preparation for mHM" in subsection Extract Data to the Size of a Catchment, Extact Data to the Time Period of Interest, Preparation of the LatLon Grid, Preparation of the Morphological Data (with the exception of soil map, hydrogeological map, LAI map), A possible GIS workflow (with the exception of aspect), Land Cover Data, and Table Data.

Additionally, forcing data is required. The forcing for mRM is grid-scale runoff. This runoff needs to be provided in a netcdf file that follows the input format of mHM for forcing variables. The only difference to the forcing variables is that the variable and file name can be both provided in the `mrm.nml` file in the namelist `mainconfig_mrm`.

Please see other details in Section Preparation of the Forcings.

Chapter 10

mHM Dependencies

NetCDF4

Reading and writing of input and output files requires the NetCDF4 library to be present. For example, version 4.1.2 can be downloaded from here. See also the NetCDF 4.1.2 release notes.

Chapter 11

Publications using mHM

- Thober, S., Cuntz, M., Kelbling, M., Kumar, R., Mai, J. and Samaniego, L., 2019. The multiscale Routing Model mRM v1. 0: simple river routing at resolutions from 1 to 50 km. Geosci. Model Dev. Discuss., 2019, pp.1-26, in press, https://doi.org/10.5194/gmd-2019-13
- Baroni, G., Schalge, B., Rakovec, O., Kumar, R., Schüler, L., Samaniego, L., et al. (2019). A comprehensive distributed hydrological modeling intercomparison to support process representation and data collection strategies. Water Resources Research, 55, 990–1010. https://doi.org/10.1029/2018WR023941
- Visser-Quinn, A., Beevers, L., Collet, L., Formetta, G., Smith, K., Wanders, N., Thober, S., Pan, M. and Kumar, R., 2019. Spatio-temporal analysis of compound hydro-hazard extremes across the UK. Advances in Water Resources, in press, https://doi.org/10.1016/j.advwatres.2019.05.019
- Wanders N., Thober S., Kumar R., Pan M., Sheffield J., Samaniego L., Wood E.F. (2019): Development and Evaluation of a Pan-European Multimodel Seasonal Hydrological Forecasting System, J. Hydrometeor., 20, 99–115, https://doi.org/10.1175/JHM-D-18-0040.1
- Jing, M., Heße, F., Kumar, R., Kolditz, O., Kalbacher, T. and Attinger, S., 2019. Influence of input and parameter uncertainty on the prediction of catchment-scale groundwater travel time distributions. Hydrology and Earth System Sciences, 23(1), pp.171-190, https://doi.org/10.5194/hess-23-171-2019
- Huang, S., Kumar, R., Rakovec, O., Aich, V., Wang, X., Samaniego, L., Liersch, S. and Krysanova, V., 2018. Multimodel assessment of flood characteristics in four large river basins at global warming of 1.5, 2.0 and 3.0 K above the pre-industrial level. Environmental Research Letters, 13(12), p.124005, https://doi.org/10.1088/1748-9326/aae94b
- Hanel, M., Rakovec, O., Markonis, Y., Maca, P., Samaniego, L., Kysely, J., Kumar, R. (2018): Revisiting the recent European droughts from a long-term perspective, Scientific Reports, https://doi.org/10.1038/s41598-018-27464-4
- Demirel, M.C., Mai, J., Mendiguren, G., Koch, J., Samaniego, L., Stisen, S. (2018): Combining satellite data and appropriate objective functions for improved spatial pattern performance of a distributed hydrologic model, Hydrol. Earth Syst. Sci. 22 (2), 1299 - 1315, https://www.hydrol-earth-syst-sci.net/22/1299/2018/
- Koch, J., Demirel, M. C., and Stisen, S. (2018): The SPAtial EFficiency metric (SPAEF): multiple-component evaluation of spatial patterns for optimization of hydrological models, Geosci. Model Dev., 11, 1873-1886, https://doi.org/10.5194/gmd-11-1873-2018/
- Höllering, S., Wienhöfer, J., Ihringer, J., Samaniego, L., Zehe, E. (2018): Regional analysis of parameter sensitivity for simulation of streamflow and hydrological fingerprints, Hydrol. Earth Syst. Sci. 22 (1), 203 - 220, https://www.hydrol-earth-syst-sci.net/22/203/2018/
- Jing, M., Heße, F., Kumar, R., Wang, W., Fischer, T., Walther, M., Zink, M., Zech, A., Samaniego, L., Kolditz, O., Attinger, S. (2018): Improved regional-scale groundwater representation by the coupling of the mesoscale Hydrologic Model (mHM v5.7) to the groundwater model OpenGeoSys (OGS), Geosci. Model Dev. 11 (5), 1989 - 2007, https://www.geosci-model-dev.net/11/1989/2018/

- Peichl, M., Thober, S., Meyer, V., Samaniego, L. (2018): The effect of soil moisture anomalies on maize yield in Germany, Nat. Hazards Earth Syst. Sci. 18 (3), 889 - 906, https://www.nat-hazards-earth-syst-sci.net/18/889/2018/

- Samaniego, L., Thober, S., Kumar, R., Wanders, N., Rakovec, O., Pan, M., Zink, M., Sheffield, J., Wood, E.F., Marx, A. (2018): Anthropogenic warming exacerbates European soil moisture droughts, Nat. Clim. Chang. 8 (5), 421 - 426, https://www.nature.com/articles/s41558-018-0138-5

- Marx, A., Kumar, R., Thober, S., Rakovec, O., Wanders, N., Zink, M., Wood, E.F., Pan, M., Sheffield, J., Samaniego, L. (2018): Climate change alters low flows in Europe under global warming of 1.5, 2, and 3 degC, Hydrol. Earth Syst. Sci. 22 (2), 1017 - 1032, https://www.hydrol-earth-syst-sci.net/22/1017/2018/

- Schrön, M., Rosolem, R., Köhli, M., Piussi, L., Schröter, I., Iwema, J., Kögler, S., Oswald, S.E., Wollschläger, U., Samaniego, L., Dietrich, P., Zacharias, S. (2018): Cosmic-ray neutron rover surveys of field soil moisture and the influence of roads, Water Resour. Res., https://doi.org/10.1029/2017WR021719

- Thober, S., Kumar, R., Wanders, N., Marx, A., Pan, M., Rakovec, O., Samaniego, L., Sheffield, J., Wood, E.F., Zink, M. (2018): Multi-model ensemble projections of European river floods and high flows at 1.5, 2, and 3 degrees global warming, Environ. Res. Lett. 13 (1), art. 014003, http://iopscience.iop.org/article/10.1088/1748-9326/aa9e35

- Zink, M., Mai, J., Cuntz, M., Samaniego, L. (2018): Conditioning a hydrologic model using patterns of remotely sensed land surface temperature, Water Resour. Res. 54 (4), 2976 - 2998, https://doi.org/10.1002/2017WR021346

- Samaniego, L., Kumar, R., Thober, S., Rakovec, O., Zink, M., Wanders, N., Eisner, S., Müller Schmied, H., Sutanudjaja, E.H., Warrach-Sagi, K., Attinger, S., (2017): Toward seamless hydrologic predictions across spatial scales, Hydrol. Earth Syst. Sci. 21 (9), 4323 - 4346, https://www.hydrol-earth-syst-sci.net/21/4323/2017/

- Samaniego, L., Kumar, R., Breuer, L., Chamorro, A., Flörke, M., Pechlivanidis, I.G., Schäfer, D., Shah, H., Vetter, T., Wortmann, M., Zeng, X., (2017): Propagation of forcing and model uncertainties on to hydrological drought characteristics in a multi-model century-long experiment in large river basins, Clim. Change 141 (3), 435 - 449, https://link.springer.com/article/10.1007/s10584-016-1778-y

- Vigiak, O., Lutz, S., Mentzafou, A., Chiogna, G., Ye, T., Majone, B., Beck, H., de Roo, A., Malagó, A., Bouraoui, F., Kumar, R., Samaniego, L., Merz, R., Gamvroudis, C., Skoulikidis, N., Nikolaidis, N.P., Bellin, A., Acuňa, V., Mori, N., Ludwig, R., Pistocchi, A., (2018): Uncertainty of modelled flow regime for flow-ecological assessment in Southern Europe, Sci. Total Environ. 615 , 1028 - 1047, https://www.sciencedirect.com/science/article/pii/S0048969717326475

- Eisner, S., Flörke, M., Chamorro, A., Daggupati, P., Donnelly, C., Huang, J., Hundecha, Y., Koch, H., Kalugin, A., Krylenko, I., Mishra, V., Piniewski, M., Samaniego, L., Seidou, O., Wallner, M., Krysanova, V., (2017): An ensemble analysis of climate change impacts on streamflow seasonality across 11 large river basins, Clim. Change 141 (3), 401 - 417, https://link.springer.com/article/10.1007/s10584-016-1844-5

- Pechlivanidis, I.G., Arheimer, B., Donnelly, C., Hundecha, Y., Huang, S., Aich, V., Samaniego, L., Eisner, S. and Shi, P., (2017): Analysis of hydrological extremes at different hydro-climatic regimes under present and future conditions. Climatic Change, 141(3), pp.467-481, https://link.springer.com/article/10.1007/s10584-016-1723-0

- Hattermann, F.F., Krysanova, V., Gosling, S.N., Dankers, R., Daggupati, P., Donnelly, C., Flörke, M., Huang, S., Motovilov, Y., Buda, S., Yang, T., Müller, C., Leng, G., Tang, Q., Portmann, F.T., Hagemann, S., Gerten, D., Wada, Y., Masaki, Y., Alemayehu, T., Satoh, Y., Samaniego, L., (2017): Cross-scale intercomparison of climate change impacts simulated by regional and global hydrological models in eleven large river basins, Clim. Change 141 (3), 561 - 576, https://link.springer.com/article/10.1007/s10584-016-1829-4

- Hattermann, F.F., Vetter, T., Breuer, L., Su, B., Daggupati, P., Donnelly, C., Fekete, B., Flörke, F., Gosling, S.N., Hoffmann, P., Liersch, S., Masaki, Y., Motovilov, Y., Müller, C., Samaniego, L., Stacke, T., Wada,

Y., Yang, T., Krysnaova, V., (2017): Sources of uncertainty in hydrological climate impact assessment : a cross-scale study, Environ. Res. Lett., http://iopscience.iop.org/article/10.1088/1748-9326/aa9938/meta

- Mishra, V., Kumar, R., Shah, H.L., Samaniego, L., Eisner, S., Yang, T., (2017): Multimodel assessment of sensitivity and uncertainty of evapotranspiration and a proxy for available water resources under climate change, Clim. Change 141 (3), 451 - 465, https://link.springer.com/article/10.1007/s10584-016-1886-8

- Zink, M., Kumar, R., Cuntz, M., & Samaniego, L. (2017). A high-resolution dataset of water fluxes and states for Germany accounting for parametric uncertainty. Hydrology and Earth System Sciences, 21(3), 1769–1790. doi:10.5194/hess-21-1769-2017, http://www.hydrol-earth-syst-sci.net/21/1769/2017/hess-21-1769-2017.html

- Heße, F., Zink, M., Kumar, R., Samaniego, L., & Attinger, S. (2017). Spatially distributed characterization of soil-moisture dynamics using travel-time distributions. Hydrology and Earth System Sciences, 21(1), 549–570. doi:10.5194/hess-21-549-2017, http://www.hydrol-earth-syst-sci.net/21/549/2017/

- Baroni, G., Zink, M., Kumar, R., Samaniego, L., & Attinger, S. (2017). Effects of uncertainty in soil properties on simulated hydrological states and fluxes at different spatio-temporal scales. Hydrology and Earth System Sciences, 21(5), 2301–2320. doi:10.5194/hess-21-2301-2017, http://www.hydrol-earth-syst-sci.net/21/2301/2017/hess-21-2301-2017.html

- Wollschlaeger, U., Attinger, S., Borchardt, D., Brauns, M., Cuntz, M., Dietrich, P., ... Zacharias, S. (2017). The Bode hydrological observatory: a platform for integrated, interdisciplinary hydro-ecological research within the TERENO Harz/Central German Lowland Observatory. Environmental Earth Sciences, 76(1), 29. doi:10.1007/s12665-016-6327-5, https://link.springer.com/article/10.1007/s12665-016-6327-5

- Mueller, C., Zink, M., Samaniego, L., Krieg, R., Merz, R., Rode, M., & Knoeller, K. (2016). Discharge Driven Nitrogen Dynamics in a Mesoscale River Basin As Constrained by Stable Isotope Patterns. Environmental Science & Technology, 50(17), 9187–9196. doi:10.1021/acs.est.6b01057, http://pubs.acs.org/doi/abs/10.1021/acs.est.6b01057

- Zink, M., Samaniego, L., Kumar, R., Thober, S., Mai, J., Schaefer, D., & Marx, A. (2016). The German drought monitor. Environmental Research Letters, 11(7), 74002. doi:10.1088/1748-9326/11/7/074002, http://iopscience.iop.org/article/10.1088/1748-9326/11/7/074002/meta

- Marx, A., Samaniego, L., Kumar, R., Thober, S., Mai, J., & Zink, M. (2016). Der Duerremonitor - Aktuelle Information zur Bodenfeuchte in Deutschland. In Wasserressourcen - Wissen in Flussgebieten vernetzen (pp. 131–142). Forum fuer Hydrologie und Wasserbewirtschaftung, http://www.ufz.de/index.php?en=20939&ufzPublicationIdentifier=17277

- Rakovec, O., Kumar, R., Mai, J., Cuntz, M., Thober, S., Zink, M., Attinger, S., Schaefer, D., Schroen, M., Samaniego, L. (2016). Multiscale and Multivariate Evaluation of Water Fluxes and States over European River Basins. Journal of Hydrometeorology, 17(1), 287–307. doi:10.1175/JHM-D-15-0054.1, http://journals.ametsoc.org/doi/abs/10.1175/JHM-D-15-0054.1

- Rakovec, O., Kumar, R., Attinger, S. and Samaniego, L. (2016). Improving the realism of hydrologic model functioning through multivariate parameter estimation. Water Resources Research, 52. doi:10.1002/2016WR019430, http://onlinelibrary.wiley.com/doi/10.1002/2016WR019430/full

- Nijzink, R. C., Samaniego, L., Mai, J., Kumar, R., Thober, S., Zink, M., Schaefer, D., Savenije, H. H. G., and Hrachowitz, M.: The importance of topography-controlled sub-grid process heterogeneity and semi-quantitative prior constraints in distributed hydrological models, Hydrol. Earth Syst. Sci., 20, 1151-1176, doi:10.5194/hess-20-1151-2016, 2016., http://www.hydrol-earth-syst-sci.net/20/1151/2016/

- Thober, S., Kumar, R., Sheffield, J., Mai, J., Schaefer, D., & Samaniego, L. (2015). Seasonal Soil Moisture Drought Prediction over Europe Using the North American Multi-Model Ensemble (NMME). Journal of Hydrometeorology, 16(6), 2329–2344. doi:10.1175/JHM-D-15-0053.1, http://journals.ametsoc.org/doi/abs/10.1175/JHM-D-15-0053.1

- Cuntz, M., Mai, J., Zink, M., Thober, S., Kumar, R., Schaefer, D., ... Samaniego, L. (2015). Computationally inexpensive identification of noninformative model parameters by sequential screening. Water Resources Research, 51(8), 6417–6441. doi:10.1002/2015WR016907, http://onlinelibrary.wiley.com/doi/10.1002/2015WR016907/full

- Livneh, B., Kumar, R., & Samaniego, L. (2015). Influence of soil textural properties on hydrologic fluxes in the Mississippi river basin. Hydrological Processes, 29(21), 4638-4655, http://onlinelibrary.wiley.com/doi/10.1002/hyp.10601/full

- Schoeniger, A., Woehling, T., Samaniego, L., & Nowak, W. (2014). Model selection on solid ground: Rigorous comparison of nine ways to evaluate Bayesian model evidence. Water resources research, 50(12), 9484-9513, http://onlinelibrary.wiley.com/doi/10.1002/2014WR016062/full

- Woehling, T., Samaniego, L., & Kumar, R. (2013). Evaluating multiple performance criteria to calibrate the distributed hydrological model of the upper Neckar catchment. Environmental Earth Sciences, 69(2), 453–468. doi:10.1007/s12665-013-2306-2, https://link.springer.com/article/10.1007/s12665-013-2306-2

- Samaniego, L., Kumar, R., & Zink, M. (2013). Implications of Parameter Uncertainty on Soil Moisture Drought Analysis in Germany. Journal of Hydrometeorology, 14(1), 47–68. doi:10.1175/JHM-D-12-075.1, http://journals.ametsoc.org/doi/abs/10.1175/JHM-D-12-075.1

- Kumar, R., Livneh, B., & Samaniego, L. (2013). Toward computationally efficient large-scale hydrologic predictions with a multiscale regionalization scheme. Water Resources Research, 49(9), 5700–5714. doi:10.1002/wrcr.20431, http://onlinelibrary.wiley.com/doi/10.1002/wrcr.20431/full

- Kumar, R., Samaniego, L., & Attinger, S. (2013). Implications of distributed hydrologic model parameterization on water fluxes at multiple scales and locations. Water Resources Research, 49(1), 360–379. doi:10.1029/2012WR012195, http://onlinelibrary.wiley.com/doi/10.1029/2012WR012195/abstract

- Zacharias, S., Bogena, H., Samaniego, L., Mauder, M., Fuß, R., Puetz, T., ... & Bens, O. (2011). A network of terrestrial environmental observatories in Germany. Vadose Zone Journal, 10(3), 955-973, https://dl.sciencesocieties.org/publications/vzj/abstracts/10/3/955

- Kalbacher, T., Delfs, J. O., Shao, H., Wang, W., Walther, M., Samaniego, L., ... & Sun, F. (2012). The IWAS-ToolBox: software coupling for an integrated water resources management. Environmental Earth Sciences, 65(5), 1367-1380, https://link.springer.com/article/10.1007/s12665-011-1270-y

- Samaniego, L., Kumar, R., & Jackisch, C. (2011). Predictions in a data-sparse region using a regionalized grid-based hydrologic model driven by remotely sensed data. Hydrology Research, 42(5), 338–355. doi:10.2166/nh.2011.156, http://hr.iwaponline.com/content/42/5/338.article-info

- Kumar, R., Samaniego, L., & Attinger, S. (2010). The effects of spatial discretization and model parameterization on the prediction of extreme runoff characteristics. Journal of Hydrology, 392(1-2), 54–69. doi:10.1016/j.jhydrol.2010.07.047, http://www.sciencedirect.com/science/article/pii/S0022169410004865

- Samaniego, L., Kumar, R., & Attinger, S. (2010). Multiscale parameter regionalization of a grid-based hydrologic model at the mesoscale. Water Resources Research, 46(5), W05523. doi:10.1029/2008WR007327, http://onlinelibrary.wiley.com/doi/10.1029/2008WR007327/full

Chapter 12

mHM RELEASE NOTES

mHM v5.10 (June 2019)

New Features:

- New routing process introduced `processCase(8) = 3` see Thober et al, 2019, GMDD, in press for more details
- mRM is decoupled from mHM and mRM now resides in `deps/mrm` as an independent submodule more information on handling submodules
- New option to compile mHM with cmake is provided, see more details under cmake manual.
- Visualization/animation R script (producing PDF and GIF) of mHM netcdf files included under post-proc `animate1.R`
- New option for coupling of mRM to a groundwater model (`gw_coupling = .true.`). The river head can be computed based on the Manning equation.

Bugs resolved from release 5.9:

- Enable use of i8 for time_data in common/mo_read_forcing_nc.f90, otherwise netcdf time stamps with initial dates prior to 1900 were wrong (due to overflow)

Known bugs:

- Adaptive routing does not allow to run without at least 1 gauge specification
- Incompatibility of Finalparam.nml format between Intel and GNU

Restrictions:

- For `gfortran` compilers mHM supports only v4.8 and higher.
- If you wish to use features connected to ground albedo neutrons (`processCase(9)`), please contact Martin Schrön.

mHM v5.9 (July 2018)

New Features:

- Major restructuralization of the mHM code.

- MPR is now executed before mHM is run.
- MPR can be compiled as a standalone tool.
- mHM without mRM can be compiled.
- The code in general is strictly reorganized into modules that belong to their respective processes (MPR, mHM, mRM). This is not only done for constants, global variables and so on, but also for every module and the reading of input as well as the namelists.
- This leads to the creation of those folders:
  - `./common` (code shared by MPR, mHM and mRM) included for every compilation option
  - `./lib` (code shared by MPR, mHM and mRM) included for every compilation option
  - `./MPR` (code for MPR), not included for mRM standalone
  - `./common_mHM_mRM` (code shared by mHM and mRM), not included for MPR standalone
  - `./mHM` (code for mHM), not included for MPR and mRM standalone
  - `./mRM` (code for mRM), not included for MPR standalone and mHM without routing
- Code is reformatted (indentation=2, spacing unified)
- Removed many duplicate code parts (e.g. shared between mHM and mRM)
- Check cases are minimized (reduced output, shorter time periods (<=2 years), less basins)
- Check cases can now be set up more easily (by use of model_wrapper for automatic creation of new nml files)
- Check cases now run a python script for output comparison, advantage: tolerance now also allowed for ascii-output and support of 4-D netCDF files without time dimension
- fSealed is now an effective parameter
- mHM effective parameters now all have three dimensions internally (nCells_L1, [iHorizon, LAI-Time], nLCoverScenes)
- Introduction of new derived types:
  - `Grid` (merge of basin_info, basin_info_mrm, gridGeoRef, nCells, longitude, latitude, Id) used for each level (0, 1, 11, 2) individually
  - `GridRemapper` (merge of lower_bound, upper_bound, etc.)
- mhm_eval and mrm_eval now have common procedure interface (needed for fully flexible optimisation)
- A new post-processor can check and adapt some fields for doxygen generation
- Changes that are not backwards-compatible:
  - Restart files are restructured (now contain only the minimum required for restart):
  - MPR: effective parameters at L1 + grid information
  - mHM: effective parameters, states and fluxes at L1 + grid information
  - mRM: routing-specific parameters, states, fluxes, configs at L1/L11 + grid information
  - mhm.nml is restructured and not backwards compatible mainly due to the reorganization of modules in dependency of their processes
  - gridded LAI values are now used for all effective parameters (no fallback to LAIclasses anymore)
  - canopy height used for aerodynamic resistance is now scaled with the actual LAI timeseries and not with a dummy timeseries of intensive orchard
- Considerable improvements and reduced redundancy in estimation of an empirical distribution of slopes (sort function in the mo_startup). Example for the Australian domain (180 million L0 cells), it reduced time for sorting slope from 32hours to only 1 minute.

- mtCLIM preprocessor in pre-proc/mtCLIM, based on `mtCLIM v4.3`. This code is able to estimate humidity (vapore pressure or vapore pressure deficit) and incoming shortwave radiation based on meteorological variables (minimum and maximum air temperature, precipitation) and morphological characteristics of the underlying terrain (digital elevation model, slope, aspect).
- mHM2OGS preprocessor in pre-proc/GIS2FEM3. This code converts the triangular-wise or quadrilateral-wise recharge data from mHM into the nodal source terms of a three dimensional finite element model for `OGS`.
- Updated documentation for CYGWIN installations under Windows 7 and 10.
- Added new objective function number 31: weighted NSE (NSE is weighted with observed discharge)
- Removal of bin files and related code (only nc and ascii files are used)

Bugs resolved from release 5.8:

- Enable use of i8 for time_data in common/mo_read_forcing_nc.f90, otherwise netcdf time stamps with initial dates prior to 1900 were wrong (due to overflow)

Known bugs:

None.

Restrictions:

- For `gfortran` compilers mHM supports only v4.8 and higher.
- If you wish to use features connected to ground albedo neutrons (`processCase(9)`), please contact `Martin Schrön`.
- If you wish to use the multi-scale Routing Model as stand-alone version, please contact `Stephan Thober`.

mHM v5.8 (Dec 2017)

New Features:

- Implementation of a new process for PET correction based on LAI at PET `process(5)=-1` (Cuneyd Demirel + *GEUS* colleagues);
- Pre-processor code for SOILGRIDS data as used for the EDgE project (Rohini Kumar);
- Reduced computational time of the neutron forward model COSMIC by factors of 30–100 (Maren Kaluza);
- Compression of the netCDF output files (David Schaefer);
- Optional project description added into the `mhm.nml`

Bugs resolved from release 5.7:

- `processCase(3)=3` did not work when compiled with openMP.
- openMP declarations missing in `mo_mpr_smhorizons.f90` for the case of `iFlag_soil=1`

Known bugs:

None.

Restrictions:

- For `gfortran` compilers mHM supports only v4.8 and higher.
- If you wish to use a special process description of evapotranspiration (`processCase(4)`) please contact `Matthias Zink`.
- If you wish to use features connected to ground albedo neutrons (`processCase(9)`), please contact `Martin Schrön`.
- If you wish to use the multi-scale Routing Model as stand-alone version, please contact `Stephan Thober`.

mHM v5.7 (Jun 2017)

New Features:

- New process descriptions for the soil evapotranspiration module:
  - Field capacity dependency to root fraction coefficient (`processCase(3)=2`) – implemented by *G↩ EUS*.
  - Jarvis (1989, J. Hydrol.) evapotranspiration reduction (`processCase(3)=3`) – implemented by *G↩ EUS*.
- Use local, monthly LAI climatology instead of look-up table, i.e., `LAI_classdefinition.txt` (`time↩ Step_LAI_input=1`).
- New objective functions for model calibration
  - Calibration of mHM using catchment average evapotransipration (`opti_function=27`).
  - Calibration of mHM using soil moisture and streamflow simultaneously (`opti_function=28`).

Bugs resolved:

- Calibration using `processCase(8)=2` (routing with adaptive timestep) does properly work now.
- Streamflow output is now properly written to the NetCDF.

Known bugs:

None

Restrictions:

- For `gfortran` compilers mHM supports only v4.8 and higher.
- If you wish to use a special process description of evapotranspiration (`processCase(4)`) please contact `Matthias Zink`.
- If you wish to use features connected to ground albedo neutrons (`processCase(9)`), please contact `Martin Schrön`.
- If you wish to use the multi-scale Routing Model as stand-alone version, please contact `Stephan Thober`.

mHM v5.6 (Dec 2016)

New Features:

- **Routing extended:** Implementation of a new parametrization for the routing model (`processCase(8)=2`). This routing option is based on an adaptive time step to improve the scalability and transferability of the model as well as a significant reduction in run time. The adaptive time step is calculated as ratio of routing resolution and celerity, the latter can be given as parameter in `mhm_parameter.nml`.

Bugs resolved:

- Any model time step from 1 h to 24 h can be chosen (in releases v5.4 and v5.5 only 1 h worked properly).
- Estimation of the Hargreaves-Samani PET for high altitudes works properly know (there have been numerical issues for high latitude values).
- Reading catchment outlets from the *restart* file works now (bug appeared in v5.5).

Known bugs:

- Calibration using `processCase(8)=2` (adaptive timestep) does not work, please use `processCase(8)=1`.

Restrictions:

- For `gfortran` compilers mHM supports only v4.8 and higher.
- If you wish to use a special process description of evapotranspiration (`processCase(4)`) please contact `Matthias Zink`.
- If you wish to use features connected to ground albedo neutrons (`processCase(9)`), please contact `Martin Schrön`.
- If you wish to use the multi-scale Routing Model as stand-alone version, please contact `Stephan Thober`.

mHM v5.5 (Jun 2016)

New Features:

- Routing works on domains with multiple outlets (e.g., continental level).
- New option for providing soil data. They can be provided as predefined layers (one map per layer).
- Speed up of mHM for big domains, due to reformulations in the model start up.
- Pre-processing: new tools for i) cutting out a catchment from a existing dataset, ii) estimation of Hargreaves-Samani evapotranspiration, and iii) enlarging the grids of the input files.

Bugs resolved:

- Assigning routing parameters is done properly now.

Known bugs:

- Specifying a model time step of 24h (in `mhm.nml`) does not work, please stick with the default time step (1h)

Restrictions:

- For `gfortran` compilers mHM supports only v4.8 and higher.
- If you wish to use a special process description of evapotranspiration (`processCase(4)`) please contact Matthias Zink.
- If you wish to use features connected to ground albedo neutrons (`processCase(9)`), please contact Martin Schrön.
- If you wish to use the multi-scale Routing Model as stand-alone version, please contact Stephan Thober.

mHM v5.4 (Dec 2015)

New Features:

- The routing of mHM can be used as a stand alone version/independent software called *multiscale Routing Model mRM*, e.g. for coupling to other environmental models.
- A new output, i.e. fields of routed discharge, is now available. They are stored in `mRM_fluxes_and_states.nc` and are controlled by a new namelist contained in the `mrm_outputs.nml` file, which is an optional file.
- New calibration objectives have been incorporated. It is now possible, additionally to the former objectives, to calibrate mHM against additional input data:
  - total water storage (e.g. GRACE) and discharge simultaneously (`opti_function=15`), and/or
  - cosmic ray neutron counts (`opti_function=17`).
- New post-processing: a mHM python class for reading all inputs and outputs of a model run can be found in `post-proc/`.
- Reorganization of the NetCDF writing in mHM to simply future implementations of additional outputs.

Bugs resolved:

- Calibration with catchment average soil moisture (`opti_function=10`) works properly now.
- Discharge output for multi basin runs with different time periods for each basin works properly now.
- Hargreaves-Samani PET calculation (`processCase(5)=1`) is valid on southern hemisphere too now.

Known bugs:

- None.

Restrictions:

- For `gfortran` compilers mHM supports only v4.8 and higher.
- If you wish to use a special process description of evapotranspiration (`processCase(4)`) please contact Matthias Zink.
- If you wish to use features connected to ground albedo neutrons (`processCase(9)`), please contact Martin Schrön.
- If you wish to use the multi-scale Routing Model as stand-alone version, please contact Stephan Thober.

mHM v5.3 (Jun 2015)

New Features:

- Simulation period and warming days can be now given per basin (see `time_periods` in `mhm.nml`)
- Enabling use of MPI (set `mpi=true` in Makefile and use `#ifdef MPI` for MPI specific code)
- Optional input data can be loaded for example to calibrate against soil moisture (see `optional_data` in `mhm.nml`)
- Generation of ground albedo cosmic-ray neutrons (see `processCase(10)` in `mhm.nml`); these calculations are based on the `COSMIC` code, which was originally written by Rafael Rosolem. Please contact `Martin Schrön` if you like to use this new feature.
- Several new objective functions, e.g. calibrating the Kling-Gupta efficiency of catchment's average soil moisture (`opti_function=10`) or calibrating multiple basins regarding Kling-Gupta efficiency of discharge (`opti_function=14`) among others; calibration against soil moisture is still purpose to research (`opti_function=10-14`). The interested user may contact `Matthias Zink` for further details.

Bugs resolved:

- Calibration using potential evapotranspiration from input file (i.e. `processCase(4)=0`) is now working properly

Known bugs:

- Compiling mHM with the recent Cygwin version under Windows is leading to an error message indicating circular dependencies. The reason for this is Unicode characters in some source code files. Please contact `mhm-admin@ufz.de`, if you get this error message. We will provide you the files with cleaned characters..

Restrictions:

- If you wish to use a special process description of evapotranspiration (i.e. Hargreaves-samani, Priestley-Taylor, or Penman-Monteith) please contact `Matthias Zink`. The special cases are set in `mhm.nml` (see `processCase(4)`).
- If you wish to use the new feature of calculating neutron counts please contact `Martin Schrön`. The feature can be enabled in `mhm.nml` (see `processCase(8)`).

mHM 5.2 (Dec 2014)

New Features:

- Chunk-wise reading of input data (see `timestep_model_inputs`)
- Complete revision of writing netCDF files
- Possibility to discard multi-scale parameter regionalization (MPR) calculations (see `perform_mpr`)
- Several process descriptions of evapotranspiration implemented (see `processCase(4)`): Read PET, Hargreaves-Samani, Priestley-Taylor, Penman-Monteith. Please contact `Matthias Zink` if you use one of the last three options, since the code is not under GNU Public license up to now.
- Adding routines for signature calculations of time series (see `mo_signatures`)
- New objective function for calibrating discharge with Kling-Gupta efficiency measure (KGE, see `opti_function`)
- New output variables (see `mhm_outputs.nml`)
- Sorting algorithm changed to public available library orderpack (see `mo_orderpack`)

Bugs resolved:

- Some variables in restart file where not assigned correctly
- Variables not initialized correctly

Known bugs:

- Calibration using PET values read from the input file (`processCase(5)=0`) is running, but yields wrong results due to a wrong initialization of variables. **The bug is resolved and will be released with version 5.3.**

mHM 5.1 (Jun 2014)

New Features:

- OpenMP handling of routines such as the multi-scale parameter regionalization (MPR)
- Multi-scale implementation, i.e. running mHM simultaneously in several basins with different resolutions
- Automatic check case framework, i.e. testing new implementations on their validity and back-compatibility
- Implementation of inflow gauges, i.e. feeding discharge time series from upstream areas at catchment boundaries
- File `gaugeinfo.txt` specifying gauging stations is now part of namelist `mhm.nml`
- Code is now free of Numerical Recipes proprietary code
- Can now run on a single cell (no routing performed) Hydrological modelling resolution (L1) equal to morphological input data resolution (L0) possible
- Windows compatible (with Cygwin)
- Support of regular geographic coordinate systems (e.g lat-lon) in addition to equal-area coordinate systems (UTM)

Bugs resolved:

- Initialization of states was not correct when running mHM in calibration mode.
- Calculated parameter values (`mhm_parameters.nml`) not necessarily in bound (check added).
- Aggregation/Disaggregation of meteorological data corrected.
- Forecast with mHM did not work because modelling period was restricted to discharge data period.
- Wrong mapping of evaluation discharge gauges for runs involving multiple gauges.

Known bugs:

- Print out of River network in Config File is wrong for Multi-Basin setup, i.e., the River network is always properly written for the first basin, but not properly for subsequent basins when these are either different ones or the same one with a different Hydrology or Routing resolution.
- mHM does not abort if x-axis of L0 (morphological data) and L2 (meteorological data) do not span over exactly the same range.

mHM 5.0 (Dec 2013)

New Features:

- Full modular version
- Automatic documentation by doxygen
- Running mHM for multiple basin simultaneously
- Definition of 8 major processes:
  - interception,
  - snow,
  - soil moisture,
  - direct runoff,
  - evapotranspiration,
  - interflow,
  - percolation,
  - routing
- Choice of different descriptions of processes possible
- Input in binary `*.bin` or netcdf `*.nc` format
- Various calibration routines and objective functions
- Consistent numerical precision handling of variables

Known bugs:

- None.

Chapter 13

Modules Index

13.1 Modules List

Here is a list of all modules with brief descriptions:

dummy_mpr . . . 83
dummy_mrm . . . 83
mo_anneal . . . 83
mo_append
    Append values on existing arrays . . . 87
mo_canopy_interc
    Canopy interception . . . 95
mo_common_constants
    Provides constants commonly used by mHM, mRM and MPR . . . 97
mo_common_file
    Provides file names and units for mRM . . . 100
mo_common_functions
    Provides small utility functions used by multiple parts of the code (mHM, mRM, MPR) . . . 102
mo_common_mhm_mrm_file
    Provides file names and units for mHM . . . 103
mo_common_mhm_mrm_read_config
    Reading of main model configurations . . . 104
mo_common_mhm_mrm_variables
    Provides structures needed by mHM, mRM and/or mpr . . . 108
mo_common_read_config
    Reading of main model configurations . . . 114
mo_common_read_data
    TODO: add description . . . 117
mo_common_restart
    TODO: add description . . . 119
mo_common_variables
    Provides structures needed by mHM, mRM and/or mpr . . . 122
mo_constants
    Provides computational, mathematical, physical, and file constants . . . 130
mo_corr . . . 139
mo_dds
    Dynamically Dimensioned Search (DDS) . . . 146
mo_errormeasures . . . 150
mo_file
    Provides file names and units for mHM . . . 175
mo_finish
    Finish a program gracefully . . . 177

mo_global_variables
Global variables ONLY used in reading, writing and startup . . . 179
mo_grid
TODO: add description . . . 192
mo_init_states
Initialization of all state variables of mHM . . . 198
mo_julian
Julian date conversion routines . . . 201
mo_kind
Define number representations . . . 229
mo_linfit
Fitting a straight line . . . 232
mo_mad . . . 233
mo_mcmc
This module is Monte Carlo Markov Chain sampling of a posterior parameter distribution . . . 234
mo_message
Write out concatenated strings . . . 237
mo_meteo_forcings
Prepare meteorological forcings data for mHM . . . 240
mo_mhm
Call all main processes of mHM . . . 248
mo_mhm_constants
Provides mHM specific constants . . . 254
mo_mhm_eval
Runs mhm with a specific parameter set and returns required variables, e.g. runoff . . . 258
mo_mhm_read_config
Reading of main model configurations . . . 262
mo_moment . . . 265
mo_mpr_constants
Provides MPR specific constants . . . 270
mo_mpr_eval
Runs MPR and writes to global effective parameters . . . 277
mo_mpr_file
Provides file names and units for mRM . . . 280
mo_mpr_global_variables
Global variables for mpr only . . . 285
mo_mpr_pet
TODO: add description . . . 294
mo_mpr_read_config
Read mpr config . . . 300
mo_mpr_restart
Reading and writing states, fluxes and configuration for restart of mHM . . . 302
mo_mpr_runoff
Multiscale parameter regionalization for runoff generation . . . 306
mo_mpr_smhorizons
Setting up the soil moisture horizons . . . 308
mo_mpr_soilmoist
Multiscale parameter regionalization (MPR) for soil moisture . . . 311
mo_mpr_startup
Startup procedures for mHM . . . 318
mo_mrm_constants
Provides mRM specific constants . . . 323
mo_mrm_eval
Runs mrm with a specific parameter set and returns required variables, e.g. runoff . . . 325
mo_mrm_file
Provides file names and units for mRM . . . 327
mo_mrm_global_variables
Global variables for mRM only . . . 333

mo_mrm_init
Wrapper for initializing Routing . . . 345
mo_mrm_mpr
Perform Multiscale Parameter Regionalization on Routing Parameters . . . 353
mo_mrm_net_startup
Startup drainage network for mHM . . . 358
mo_mrm_objective_function_runoff
Objective Functions for Optimization of mHM/mRM against runoff . . . 374
mo_mrm_read_config
Read mRM config . . . 400
mo_mrm_read_data
This module contains all routines to read mRM data from file . . . 404
mo_mrm_restart
Restart routines . . . 409
mo_mrm_river_head . . . 413
mo_mrm_routing
Performs runoff routing for mHM at level L11 . . . 419
mo_mrm_signatures
Module with calculations for several hydrological signatures . . . 427
mo_mrm_write
Write of discharge and restart files . . . 436
mo_mrm_write_fluxes_states
Creates NetCDF output for different fluxes and state variables of mHM . . . 446
mo_multi_param_reg
Multiscale parameter regionalization (MPR) . . . 455
mo_ncread . . . 467
mo_ncwrite . . . 487
mo_netcdf
NetCDF Fortran 90 interface wrapper . . . 515
mo_neutrons
Models to predict neutron intensities above soils . . . 579
mo_nml
Deal with namelist files . . . 588
mo_objective_function
Objective Functions for Optimization of mHM . . . 594
mo_optimization
Wrapper subroutine for optimization against runoff and sm . . . 610
mo_optimization_utils . . . 612
mo_orderpack
Sort and ranking routines . . . 613
mo_percentile . . . 630
mo_pet
Module for calculating reference/potential evapotranspiration [mm s-1] . . . 632
mo_prepare_gridded_lai
Prepare daily LAI fields (e.g., MODIS data) for mHM . . . 640
mo_read_forcing_nc
Reads forcing input data . . . 643
mo_read_latlon
Reading latitude and longitude coordinates for each basin . . . 649
mo_read_lut
Routines reading lookup tables (lut) . . . 651
mo_read_optional_data
Read optional data for mHM calibration . . . 654
mo_read_spatial_data
Reads spatial input data . . . 658
mo_read_timeseries
Routines to read files containing timeseries data . . . 662

mo_read_wrapper
Wrapper for all reading routines . . . 664
mo_restart
Reading and writing states, fluxes and configuration for restart of mHM . . . 667
mo_runoff
Runoff generation for the unsaturated zone, saturated zone (or groundwater zone), and runoff accumulation . . . 671
mo_sce
Shuffled Complex Evolution optimization algorithm . . . 675
mo_set_netcdf_outputs
Defines the structure of the netCDF to write the output in . . . 683
mo_snow_accum_melt
Snow melting and accumulation . . . 684
mo_soil_database
Generating soil database from input file . . . 685
mo_soil_moisture
Soil moisture of the different layers . . . 688
mo_spatial_agg_disagg_forcing
Spatial aggegation or disaggregation of meteorological input data . . . 692
mo_spatialsimilarity
Routines for bias insensitive comparison of spatial patterns . . . 694
mo_standard_score
Routines for calculating the normalization (anomaly)/standard score/z score and the deseasonalized (standard score on monthly basis) values of a time series . . . 695
mo_startup
Startup procedures for mHM . . . 697
mo_string_utils
String utilities . . . 701
mo_template
Template for future module developments . . . 708
mo_temporal_aggregation
Temporal aggregation for time series (averaging) . . . 710
mo_temporal_disagg_forcing
Temporal disaggregation of daily input values . . . 711
mo_timer
Timing routines . . . 713
mo_upscaling_operators
Module containing upscaling operators . . . 722
mo_utils
General utilities for the CHS library . . . 729
mo_write_ascii
Module to write ascii file output . . . 734
mo_write_fluxes_states
Creates NetCDF output for different fluxes and state variables of mHM . . . 738
mo_xor4096 . . . 747

Chapter 14

Data Type Index

14.1 Data Types List

Here are the data types with brief descriptions:

mo_moment::absdev . . . 753
mo_anneal::anneal
    Anneal . . . 753
mo_append::append
    Append (rows) scalars, vectors, and matrixes onto existing array . . . 756
mo_corr::arth . . . 761
mo_ncwrite::attribute . . . 763
mo_corr::autocoeffk . . . 764
mo_corr::autocorr . . . 765
mo_moment::average . . . 765
mo_mrm_global_variables::basininfo_mrm . . . 766
mo_errormeasures::bias . . . 768
mo_moment::central_moment . . . 770
mo_moment::central_moment_var . . . 770
mo_standard_score::classified_standard_score
    Calculates the classified standard score (e.g. classes are months) . . . 771
mo_corr::corr . . . 772
mo_moment::correlation . . . 773
mo_moment::covariance . . . 773
mo_corr::crosscoeffk . . . 774
mo_corr::crosscorr . . . 775
mo_orderpack::ctrper . . . 775
mo_temporal_aggregation::day2mon_average
    Day-to-month average (day2mon_average) . . . 776
mo_ncwrite::dims . . . 777
mo_ncwrite::dump_netcdf . . . 778
mo_utils::eq . . . 782
mo_utils::equal
    Comparison of real values . . . 782
mo_optimization_utils::eval_interface . . . 783
mo_orderpack::fndnth . . . 784
mo_corr::four1 . . . 785
mo_corr::fourrow . . . 785
mo_mrm_global_variables::gaugingstation . . . 786
mo_utils::ge . . . 787
mo_anneal::generate_neighborhood_weight . . . 787
mo_ncread::get_ncvar . . . 788
mo_xor4096::get_timeseed . . . 793

mo_anneal::gettemperature
GetTemperature . . . 794
mo_utils::greaterequal . . . 796
mo_common_variables::grid . . . 797
mo_common_variables::gridremapper . . . 800
mo_temporal_aggregation::hour2day_average
Hour-to-day average (hour2day_average) . . . 801
mo_orderpack::indmed . . . 802
mo_orderpack::indnth . . . 803
mo_orderpack::inspar . . . 804
mo_orderpack::inssor . . . 805
mo_utils::is_finite
.true. if not IEEE Inf, IEEE NaN, nor IEEE Inf nor IEEE NaN, respectively . . . 805
mo_utils::is_nan . . . 806
mo_utils::is_normal . . . 807
mo_errormeasures::kge
Kling-Gupta-Efficiency measure . . . 807
mo_errormeasures::kgenocorr
Kling-Gupta-Efficiency measure without correlation . . . 809
mo_moment::kurtosis . . . 811
mo_utils::le . . . 812
mo_utils::lesserequal . . . 813
mo_linfit::linfit
Fits a straight line to input data by minimizing chi^2 . . . 813
mo_errormeasures::lnnse . . . 814
mo_utils::locate
Find closest values in a monotonic series, returns the indexes . . . 816
mo_mad::mad . . . 817
mo_errormeasures::mae . . . 818
mo_mcmc::mcmc
This module is Monte Carlo Markov Chain sampling of a posterior parameter distribution (measurement errors are either known or modeled) . . . 819
mo_mcmc::mcmc_stddev
This module is Monte Carlo Markov Chain sampling of a posterior parameter distribution (measurement errors are neither known nor modeled) . . . 824
mo_moment::mean . . . 828
mo_template::mean
The average . . . 828
mo_percentile::median . . . 829
mo_moment::mixed_central_moment . . . 830
mo_moment::mixed_central_moment_var . . . 831
mo_moment::moment . . . 831
mo_orderpack::mrgref . . . 832
mo_orderpack::mrgrnk . . . 833
mo_errormeasures::mse . . . 833
mo_orderpack::mulcnt . . . 835
mo_percentile::n_element . . . 835
mo_netcdf::ncdataset
Provides basic file modification functionality . . . 836
mo_netcdf::ncdimension
Provides the dimension access functionality . . . 847
mo_netcdf::ncvariable . . . 867
mo_utils::ne . . . 867
mo_orderpack::nearless . . . 868
mo_spatialsimilarity::nndv
Calculates the number of neighboring dominating values, a measure for spatial dissimilarity . . . 868
mo_utils::notequal . . . 870
mo_errormeasures::nse . . . 871

mo_string_utils::num2str
Convert to string . . . 872
mo_string_utils::numarray2str
Convert to string . . . 874
mo_optimization_utils::objective_interface . . . 875
mo_orderpack::omedian . . . 875
mo_mrm_write_fluxes_states::outputdataset . . . 876
mo_write_fluxes_states::outputdataset . . . 880
mo_mrm_write_fluxes_states::outputvariable . . . 884
mo_write_fluxes_states::outputvariable . . . 889
mo_append::paste
Paste (columns) scalars, vectors, and matrixes onto existing array . . . 893
mo_spatialsimilarity::pd
Calculates pattern dissimilarity (PD) measure . . . 897
mo_percentile::percentile . . . 899
mo_common_variables::period . . . 900
mo_percentile::qmedian . . . 902
mo_orderpack::rapknr . . . 902
mo_read_spatial_data::read_spatial_data_ascii
Reads spatial data files of ASCII format . . . 903
mo_corr::realft . . . 905
mo_orderpack::refpar . . . 906
mo_orderpack::refsor . . . 907
mo_orderpack::rinpar . . . 907
mo_errormeasures::rmse . . . 908
mo_orderpack::rnkpar . . . 909
mo_errormeasures::sae . . . 910
mo_julian::setcalendar . . . 911
mo_moment::skewness . . . 912
mo_mpr_global_variables::soiltype . . . 913
mo_orderpack::sort
Unconditional ranking . . . 916
mo_orderpack::sort_index . . . 919
mo_spatial_agg_disagg_forcing::spatial_aggregation
Spatial aggregation of meterological variables . . . 920
mo_spatial_agg_disagg_forcing::spatial_disaggregation
Spatial disaggregation of meterological variables . . . 921
mo_utils::special_value
Special IEEE values . . . 922
mo_kind::sprs2_dp
Double Precision Numerical Recipes types for sparse arrays . . . 924
mo_kind::sprs2_sp
Single Precision Numerical Recipes types for sparse arrays . . . 925
mo_errormeasures::sse . . . 926
mo_standard_score::standard_score
Calculates the standard score / normalization (anomaly) / z-score . . . 928
mo_moment::stddev . . . 929
mo_corr::swap . . . 930
mo_utils::swap
Swap to values or two elements in array . . . 930
mo_global_variables::twsstructure . . . 932
mo_orderpack::uniinv . . . 933
mo_orderpack::unipar . . . 934
mo_orderpack::unirnk . . . 934
mo_orderpack::unista . . . 935
mo_restart::unpack_field_and_write
TODO: add description . . . 936

mo_mpr_restart::unpack_field_and_write
TODO: add description 937
mo_orderpack::valmed 939
mo_orderpack::valnth 940
mo_ncwrite::var2nc
Extended dump_netcdf for multiple variables 941
mo_ncwrite::variable 948
mo_moment::variance 953
mo_errormeasures::wnse 954
mo_xor4096::xor4096 955
mo_xor4096::xor4096g 957

Chapter 15

File Index

15.1 File List

Here is a list of all files with brief descriptions:

mhm_driver.f90 . . . 959
mo_anneal.f90 . . . 962
mo_append.f90 . . . 963
mo_canopy_interc.f90 . . . 964
mo_common_constants.f90 . . . 964
mo_common_file.f90 . . . 965
mo_common_functions.f90 . . . 965
mo_common_mHM_mRM_file.f90 . . . 966
mo_common_mHM_mRM_read_config.f90 . . . 966
mo_common_mHM_mRM_variables.f90 . . . 966
mo_common_read_config.f90 . . . 967
mo_common_read_data.f90 . . . 967
mo_common_restart.f90 . . . 968
mo_common_variables.f90 . . . 968
mo_constants.f90 . . . 969
mo_corr.f90 . . . 971
mo_dds.f90 . . . 972
mo_errormeasures.f90 . . . 973
mo_file.f90 . . . 974
mo_finish.f90 . . . 975
mo_global_variables.f90 . . . 975
mo_grid.f90 . . . 977
mo_init_states.f90 . . . 977
mo_julian.f90 . . . 978
mo_kind.f90 . . . 979
mo_linfit.f90 . . . 980
mo_mad.f90 . . . 980
mo_mcmc.f90 . . . 980
mo_message.f90 . . . 981
mo_meteo_forcings.f90 . . . 981
mo_mhm.f90 . . . 982
mo_mhm_constants.f90 . . . 982
mo_mhm_eval.f90 . . . 983
mo_mhm_read_config.f90 . . . 983
mo_moment.f90 . . . 983
mo_mpr_constants.f90 . . . 985
mo_mpr_eval.f90 . . . 986
mo_mpr_file.f90 . . . 986

mo_mpr_global_variables.f90 . . . 987
mo_mpr_pet.f90 . . . 988
mo_mpr_read_config.f90 . . . 989
mo_mpr_restart.f90 . . . 989
mo_mpr_runoff.f90 . . . 990
mo_mpr_smhorizons.f90 . . . 990
mo_mpr_soilmoist.f90 . . . 990
mo_mpr_startup.f90 . . . 991
mo_mrm_constants.f90 . . . 991
mo_mrm_eval.f90 . . . 991
mo_mrm_file.f90 . . . 992
mo_mrm_global_variables.f90 . . . 993
mo_mrm_init.f90 . . . 995
mo_mrm_mpr.f90 . . . 995
mo_mrm_net_startup.f90 . . . 995
mo_mrm_objective_function_runoff.f90 . . . 997
mo_mrm_read_config.f90 . . . 998
mo_mrm_read_data.f90 . . . 998
mo_mrm_restart.f90 . . . 999
mo_mrm_river_head.f90 . . . 999
mo_mrm_routing.f90 . . . 1000
mo_mrm_signatures.f90 . . . 1000
mo_mrm_write.f90 . . . 1001
mo_mrm_write_fluxes_states.f90 . . . 1001
mo_multi_param_reg.f90 . . . 1002
mo_ncread.f90 . . . 1003
mo_ncwrite.f90 . . . 1004
mo_netcdf.f90 . . . 1005
mo_neutrons.f90 . . . 1008
mo_nml.f90 . . . 1009
mo_objective_function.f90 . . . 1009
mo_optimization.f90 . . . 1010
mo_optimization_utils.f90 . . . 1010
mo_orderpack.f90 . . . 1011
mo_percentile.f90 . . . 1013
mo_pet.f90 . . . 1013
mo_prepare_gridded_lai.f90 . . . 1014
mo_read_forcing_nc.f90 . . . 1014
mo_read_latlon.f90 . . . 1015
mo_read_lut.f90 . . . 1015
mo_read_optional_data.f90 . . . 1015
mo_read_spatial_data.f90 . . . 1016
mo_read_timeseries.f90 . . . 1016
mo_read_wrapper.f90 . . . 1016
mo_restart.f90 . . . 1017
mo_runoff.f90 . . . 1017
mo_sce.f90 . . . 1017
mo_set_netcdf_outputs.f90 . . . 1020
mo_snow_accum_melt.f90 . . . 1020
mo_soil_database.f90 . . . 1020
mo_soil_moisture.f90 . . . 1021
mo_spatial_agg_disagg_forcing.f90 . . . 1021
mo_spatialsimilarity.f90 . . . 1022
mo_standard_score.f90 . . . 1022
mo_startup.f90 . . . 1022
mo_string_utils.f90 . . . 1023
mo_template.f90 . . . 1024
mo_temporal_aggregation.f90 . . . 1024

mo_temporal_disagg_forcing.f90 1025
mo_timer.f90 1025
mo_upscaling_operators.f90 1026
mo_utils.f90 1026
mo_write_ascii.f90 1027
mo_write_fluxes_states.f90 1028
mo_xor4096.f90 1029
mpr_driver.f90 1029
mrm_driver.f90 1031

Chapter 16

Module Documentation

16.1 dummy_mpr Module Reference

16.2 dummy_mrm Module Reference

16.3 mo_anneal Module Reference

Data Types

- interface anneal

  anneal
- interface generate_neighborhood_weight
- interface gettemperature

  GetTemperature.

Functions/Subroutines

- real(dp) function, dimension(size(para, 1)) anneal_dp (eval, cost, para, prange, prange_func, temp, Dt, nl↩ TERmax, Len, nST, eps, acc, seeds, printflag, maskpara, weight, changeParaMode, reflectionFlag, pertub↩ FlexFlag, maxit, undef_funcval, tmp_file, funcbest, history)
- real(dp) function gettemperature_dp (paraset, cost, eval, acc_goal, prange, prange_func, samplesize, maskpara, seeds, printflag, weight, maxit, undef_funcval)
- real(dp) function pargen_anneal_dp (old, dMax, oMin, oMax, RN)
- real(dp) function pargen_dds_dp (old, perturb, oMin, oMax, RN)
- real(dp) function dchange_dp (delta, iDigit, isZero)
- subroutine generate_neighborhood_weight_dp (truepara, cum_weight, save_state_xor, iTotalCounter, nIT↩ ERmax, neighborhood)

16.3.1 Function/Subroutine Documentation

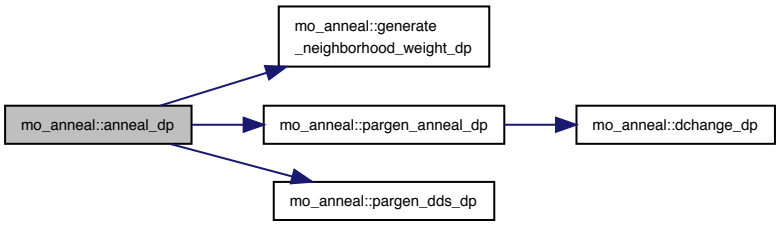
16.3.1.1 anneal_dp()

```
real(dp) function, dimension(size(para, 1)) mo_anneal::anneal_dp (
            procedure(eval_interface), intent(in), pointer eval,
            procedure(objective_interface), intent(in), pointer cost,
```

```
            real(dp), dimension(:), intent(in) para,
            real(dp), dimension(size(para, 1), 2), intent(in), optional prange,
            optional prange_func,
            real(dp), intent(in), optional temp,
            real(dp), intent(in), optional Dt,
            integer(i4), intent(in), optional nITERmax,
            integer(i4), intent(in), optional Len,
            integer(i4), intent(in), optional nST,
            real(dp), intent(in), optional eps,
            real(dp), intent(in), optional acc,
            integer(i8), dimension(3), intent(in), optional seeds,
            logical, intent(in), optional printflag,
            logical, dimension(size(para, 1)), intent(in), optional maskpara,
            real(dp), dimension(size(para, 1)), intent(in), optional weight,
            integer(i4), intent(in), optional changeParaMode,
            logical, intent(in), optional reflectionFlag,
            logical, intent(in), optional pertubFlexFlag,
            logical, intent(in), optional maxit,
            real(dp), intent(in), optional undef_funcval,
            character(len = *), intent(in), optional tmp_file,
            real(dp), intent(out), optional funcbest,
            real(dp), dimension(:, :), intent(out), optional, allocatable history )  [private]
```

References generate_neighborhood_weight_dp(), pargen_anneal_dp(), and pargen_dds_dp().

Here is the call graph for this function:



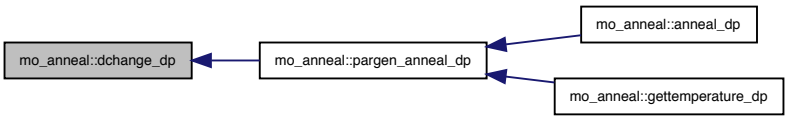
16.3.1.2 dchange_dp()

```
real(dp) function mo_anneal::dchange_dp (
            real(dp), intent(in) delta,
            integer(i4), intent(in) iDigit,
            integer(i4), intent(in) isZero )  [private]
```

References mo_kind::dp, mo_kind::i4, and mo_kind::i8.

Referenced by pargen_anneal_dp().

Here is the caller graph for this function:

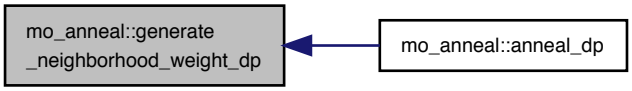


16.3.1.3 generate_neighborhood_weight_dp()

```
subroutine mo_anneal::generate_neighborhood_weight_dp (
             integer(i4), dimension(:), intent(in) truepara,
             real(dp), dimension(:), intent(in) cum_weight,
             integer(i8), dimension(n_save_state), intent(inout) save_state_xor,
             integer(i4), intent(in) iTotalCounter,
             integer(i4), intent(in) nITERmax,
             logical, dimension(size(cum_weight)), intent(out) neighborhood )
```

Referenced by anneal_dp().

Here is the caller graph for this function:

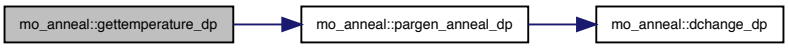


16.3.1.4 gettemperature_dp()

```
real(dp) function mo_anneal::gettemperature_dp (
             real(dp), dimension(:), intent(in) paraset,
             procedure(objective_interface), intent(in), pointer cost,
             procedure(eval_interface), intent(in), pointer eval,
             real(dp), intent(in) acc_goal,
             real(dp), dimension(size(paraset, 1), 2), intent(in), optional prange,
             optional prange_func,
             integer(i4), intent(in), optional samplesize,
             logical, dimension(size(paraset, 1)), intent(in), optional maskpara,
             integer(i8), dimension(2), intent(in), optional seeds,
             logical, intent(in), optional printflag,
             real(dp), dimension(size(paraset, 1)), intent(in), optional weight,
             logical, intent(in), optional maxit,
             real(dp), intent(in), optional undef_funcval )
```

References mo_kind::dp, mo_kind::i4, mo_kind::i8, and pargen_anneal_dp().

Here is the call graph for this function:



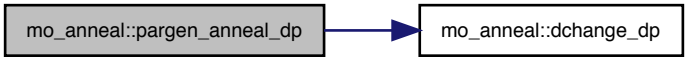
16.3.1.5 pargen_anneal_dp()

```
real(dp) function mo_anneal::pargen_anneal_dp (
             real(dp), intent(in) old,
             real(dp), intent(in) dMax,
             real(dp), intent(in) oMin,
             real(dp), intent(in) oMax,
             real(dp), intent(in) RN )
```

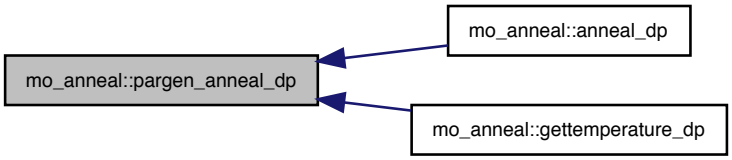
References dchange_dp(), mo_kind::dp, and mo_kind::i4.

Referenced by anneal_dp(), and gettemperature_dp().

Here is the call graph for this function:



Here is the caller graph for this function:

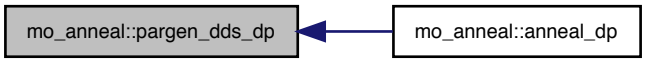


16.3.1.6 pargen_dds_dp()

```
real(dp) function mo_anneal::pargen_dds_dp (
             real(dp), intent(in) old,
             real(dp), intent(in) perturb,
             real(dp), intent(in) oMin,
             real(dp), intent(in) oMax,
             real(dp), intent(in) RN )
```

Referenced by anneal_dp().

Here is the caller graph for this function:



16.4 mo_append Module Reference

Append values on existing arrays.

Data Types

- interface append

  Append (rows) scalars, vectors, and matrixes onto existing array.
- interface paste

  Paste (columns) scalars, vectors, and matrixes onto existing array.

Functions/Subroutines

- subroutine append_i4_v_s (vec1, sca2)
- subroutine append_i4_v_v (vec1, vec2)
- subroutine append_i4_m_m (mat1, mat2, fill_value)
- subroutine append_i4_3d (mat1, mat2, fill_value)
- subroutine append_i8_v_s (vec1, sca2)
- subroutine append_i8_v_v (vec1, vec2)
- subroutine append_i8_m_m (mat1, mat2, fill_value)
- subroutine append_i8_3d (mat1, mat2, fill_value)
- subroutine append_sp_v_s (vec1, sca2)
- subroutine append_sp_v_v (vec1, vec2)
- subroutine append_sp_m_m (mat1, mat2, fill_value)
- subroutine append_sp_3d (mat1, mat2, fill_value)
- subroutine append_dp_v_s (vec1, sca2)
- subroutine append_dp_v_v (vec1, vec2)
- subroutine append_dp_m_m (mat1, mat2, fill_value)
- subroutine append_dp_3d (mat1, mat2, fill_value)
- subroutine append_char_v_s (vec1, sca2)
- subroutine append_char_v_v (vec1, vec2)

- subroutine append_char_m_m (mat1, mat2, fill_value)
- subroutine append_char_3d (mat1, mat2, fill_value)
- subroutine append_lgt_v_s (vec1, sca2)
- subroutine append_lgt_v_v (vec1, vec2)
- subroutine append_lgt_m_m (mat1, mat2, fill_value)
- subroutine append_lgt_3d (mat1, mat2, fill_value)
- subroutine paste_i4_m_s (mat1, sca2)
- subroutine paste_i4_m_v (mat1, vec2, fill_value)
- subroutine paste_i4_m_m (mat1, mat2, fill_value)
- subroutine paste_i8_m_s (mat1, sca2)
- subroutine paste_i8_m_v (mat1, vec2, fill_value)
- subroutine paste_i8_m_m (mat1, mat2, fill_value)
- subroutine paste_sp_m_s (mat1, sca2)
- subroutine paste_sp_m_v (mat1, vec2, fill_value)
- subroutine paste_sp_m_m (mat1, mat2, fill_value)
- subroutine paste_dp_m_s (mat1, sca2)
- subroutine paste_dp_m_v (mat1, vec2, fill_value)
- subroutine paste_dp_m_m (mat1, mat2, fill_value)
- subroutine paste_char_m_s (mat1, sca2)
- subroutine paste_char_m_v (mat1, vec2, fill_value)
- subroutine paste_char_m_m (mat1, mat2, fill_value)
- subroutine paste_lgt_m_s (mat1, sca2)
- subroutine paste_lgt_m_v (mat1, vec2)
- subroutine paste_lgt_m_m (mat1, mat2)

16.4.1 Detailed Description

Append values on existing arrays.

Provides routines to append (rows) and paste (columns) scalars, vectors, and matrixes onto existing arrays.

Author

Juliane Mai

Date

Aug 2012

16.4.2 Function/Subroutine Documentation

16.4.2.1 append_char_3d()

```
subroutine mo_append::append_char_3d (
            character(len = *), dimension(:, :, :), intent(inout), allocatable mat1,
            character(len = *), dimension(:, :, :), intent(in) mat2,
            character(len = *), intent(in), optional fill_value )  [private]
```

16.4.2.2 append_char_m_m()

```
subroutine mo_append::append_char_m_m (
             character(len = *), dimension(:, :), intent(inout), allocatable mat1,
             character(len = *), dimension(:, :), intent(in) mat2,
             character(len = *), intent(in), optional fill_value )  [private]
```

16.4.2.3 append_char_v_s()

```
subroutine mo_append::append_char_v_s (
             character(len = *), dimension(:), intent(inout), allocatable vec1,
             character(len = *), intent(in) sca2 )  [private]
```

16.4.2.4 append_char_v_v()

```
subroutine mo_append::append_char_v_v (
             character(len = *), dimension(:), intent(inout), allocatable vec1,
             character(len = *), dimension(:), intent(in) vec2 )  [private]
```

16.4.2.5 append_dp_3d()

```
subroutine mo_append::append_dp_3d (
             real(dp), dimension(:, :, :), intent(inout), allocatable mat1,
             real(dp), dimension(:, :, :), intent(in) mat2,
             real(dp), intent(in), optional fill_value )  [private]
```

16.4.2.6 append_dp_m_m()

```
subroutine mo_append::append_dp_m_m (
             real(dp), dimension(:, :), intent(inout), allocatable mat1,
             real(dp), dimension(:, :), intent(in) mat2,
             real(dp), intent(in), optional fill_value )  [private]
```

16.4.2.7 append_dp_v_s()

```
subroutine mo_append::append_dp_v_s (
             real(dp), dimension(:), intent(inout), allocatable vec1,
             real(dp), intent(in) sca2 )  [private]
```

16.4.2.8 append_dp_v_v()

```
subroutine mo_append::append_dp_v_v (
             real(dp), dimension(:), intent(inout), allocatable vec1,
             real(dp), dimension(:), intent(in) vec2 )  [private]
```

16.4.2.9 append_i4_3d()

```
subroutine mo_append::append_i4_3d (
            integer(i4), dimension(:, :, :), intent(inout), allocatable mat1,
            integer(i4), dimension(:, :, :), intent(in) mat2,
            integer(i4), intent(in), optional fill_value )  [private]
```

16.4.2.10 append_i4_m_m()

```
subroutine mo_append::append_i4_m_m (
            integer(i4), dimension(:, :), intent(inout), allocatable mat1,
            integer(i4), dimension(:, :), intent(in) mat2,
            integer(i4), intent(in), optional fill_value )  [private]
```

16.4.2.11 append_i4_v_s()

```
subroutine mo_append::append_i4_v_s (
            integer(i4), dimension(:), intent(inout), allocatable vec1,
            integer(i4), intent(in) sca2 )  [private]
```

16.4.2.12 append_i4_v_v()

```
subroutine mo_append::append_i4_v_v (
            integer(i4), dimension(:), intent(inout), allocatable vec1,
            integer(i4), dimension(:), intent(in) vec2 )  [private]
```

16.4.2.13 append_i8_3d()

```
subroutine mo_append::append_i8_3d (
            integer(i8), dimension(:, :, :), intent(inout), allocatable mat1,
            integer(i8), dimension(:, :, :), intent(in) mat2,
            integer(i8), intent(in), optional fill_value )  [private]
```

16.4.2.14 append_i8_m_m()

```
subroutine mo_append::append_i8_m_m (
            integer(i8), dimension(:, :), intent(inout), allocatable mat1,
            integer(i8), dimension(:, :), intent(in) mat2,
            integer(i8), intent(in), optional fill_value )  [private]
```

16.4.2.15 append_i8_v_s()

```
subroutine mo_append::append_i8_v_s (
             integer(i8), dimension(:), intent(inout), allocatable vec1,
             integer(i8), intent(in) sca2 )  [private]
```

16.4.2.16 append_i8_v_v()

```
subroutine mo_append::append_i8_v_v (
             integer(i8), dimension(:), intent(inout), allocatable vec1,
             integer(i8), dimension(:), intent(in) vec2 )  [private]
```

16.4.2.17 append_lgt_3d()

```
subroutine mo_append::append_lgt_3d (
             logical, dimension(:, :, :), intent(inout), allocatable mat1,
             logical, dimension(:, :, :), intent(in) mat2,
             logical, intent(in), optional fill_value )  [private]
```

16.4.2.18 append_lgt_m_m()

```
subroutine mo_append::append_lgt_m_m (
             logical, dimension(:, :), intent(inout), allocatable mat1,
             logical, dimension(:, :), intent(in) mat2,
             logical, intent(in), optional fill_value )  [private]
```

16.4.2.19 append_lgt_v_s()

```
subroutine mo_append::append_lgt_v_s (
             logical, dimension(:), intent(inout), allocatable vec1,
             logical, intent(in) sca2 )  [private]
```

16.4.2.20 append_lgt_v_v()

```
subroutine mo_append::append_lgt_v_v (
             logical, dimension(:), intent(inout), allocatable vec1,
             logical, dimension(:), intent(in) vec2 )  [private]
```

16.4.2.21 append_sp_3d()

```
subroutine mo_append::append_sp_3d (
             real(sp), dimension(:, :, :), intent(inout), allocatable mat1,
             real(sp), dimension(:, :, :), intent(in) mat2,
             real(sp), intent(in), optional fill_value )  [private]
```

16.4.2.22 append_sp_m_m()

```
subroutine mo_append::append_sp_m_m (
             real(sp), dimension(:, :), intent(inout), allocatable mat1,
             real(sp), dimension(:, :), intent(in) mat2,
             real(sp), intent(in), optional fill_value )  [private]
```

16.4.2.23 append_sp_v_s()

```
subroutine mo_append::append_sp_v_s (
             real(sp), dimension(:), intent(inout), allocatable vec1,
             real(sp), intent(in) sca2 )  [private]
```

16.4.2.24 append_sp_v_v()

```
subroutine mo_append::append_sp_v_v (
             real(sp), dimension(:), intent(inout), allocatable vec1,
             real(sp), dimension(:), intent(in) vec2 )  [private]
```

16.4.2.25 paste_char_m_m()

```
subroutine mo_append::paste_char_m_m (
             character(len = *), dimension(:, :), intent(inout), allocatable mat1,
             character(len = *), dimension(:, :), intent(in) mat2,
             character(len = *), intent(in), optional fill_value )  [private]
```

16.4.2.26 paste_char_m_s()

```
subroutine mo_append::paste_char_m_s (
             character(len = *), dimension(:, :), intent(inout), allocatable mat1,
             character(len = *), intent(in) sca2 )  [private]
```

16.4.2.27 paste_char_m_v()

```
subroutine mo_append::paste_char_m_v (
             character(len = *), dimension(:, :), intent(inout), allocatable mat1,
             character(len = *), dimension(:), intent(in) vec2,
             character(len = *), intent(in), optional fill_value )  [private]
```

16.4.2.28 paste_dp_m_m()

```
subroutine mo_append::paste_dp_m_m (
```

```
            real(dp), dimension(:, :), intent(inout), allocatable mat1,
            real(dp), dimension(:, :), intent(in) mat2,
            real(dp), intent(in), optional fill_value )  [private]
```

16.4.2.29 paste_dp_m_s()

```
subroutine mo_append::paste_dp_m_s (
            real(dp), dimension(:, :), intent(inout), allocatable mat1,
            real(dp), intent(in) sca2 )  [private]
```

16.4.2.30 paste_dp_m_v()

```
subroutine mo_append::paste_dp_m_v (
            real(dp), dimension(:, :), intent(inout), allocatable mat1,
            real(dp), dimension(:), intent(in) vec2,
            real(dp), intent(in), optional fill_value )  [private]
```

16.4.2.31 paste_i4_m_m()

```
subroutine mo_append::paste_i4_m_m (
            integer(i4), dimension(:, :), intent(inout), allocatable mat1,
            integer(i4), dimension(:, :), intent(in) mat2,
            integer(i4), intent(in), optional fill_value )  [private]
```

16.4.2.32 paste_i4_m_s()

```
subroutine mo_append::paste_i4_m_s (
            integer(i4), dimension(:, :), intent(inout), allocatable mat1,
            integer(i4), intent(in) sca2 )  [private]
```

16.4.2.33 paste_i4_m_v()

```
subroutine mo_append::paste_i4_m_v (
            integer(i4), dimension(:, :), intent(inout), allocatable mat1,
            integer(i4), dimension(:), intent(in) vec2,
            integer(i4), intent(in), optional fill_value )  [private]
```

16.4.2.34 paste_i8_m_m()

```
subroutine mo_append::paste_i8_m_m (
            integer(i8), dimension(:, :), intent(inout), allocatable mat1,
            integer(i8), dimension(:, :), intent(in) mat2,
            integer(i8), intent(in), optional fill_value )  [private]
```

16.4.2.35 paste_i8_m_s()

```
subroutine mo_append::paste_i8_m_s (
            integer(i8), dimension(:, :), intent(inout), allocatable mat1,
            integer(i8), intent(in) sca2 )  [private]
```

16.4.2.36 paste_i8_m_v()

```
subroutine mo_append::paste_i8_m_v (
            integer(i8), dimension(:, :), intent(inout), allocatable mat1,
            integer(i8), dimension(:), intent(in) vec2,
            integer(i8), intent(in), optional fill_value )  [private]
```

16.4.2.37 paste_lgt_m_m()

```
subroutine mo_append::paste_lgt_m_m (
            logical, dimension(:, :), intent(inout), allocatable mat1,
            logical, dimension(:, :), intent(in) mat2 )  [private]
```

16.4.2.38 paste_lgt_m_s()

```
subroutine mo_append::paste_lgt_m_s (
            logical, dimension(:, :), intent(inout), allocatable mat1,
            logical, intent(in) sca2 )  [private]
```

16.4.2.39 paste_lgt_m_v()

```
subroutine mo_append::paste_lgt_m_v (
            logical, dimension(:, :), intent(inout), allocatable mat1,
            logical, dimension(:), intent(in) vec2 )  [private]
```

16.4.2.40 paste_sp_m_m()

```
subroutine mo_append::paste_sp_m_m (
            real(sp), dimension(:, :), intent(inout), allocatable mat1,
            real(sp), dimension(:, :), intent(in) mat2,
            real(sp), intent(in), optional fill_value )  [private]
```

16.4.2.41 paste_sp_m_s()

```
subroutine mo_append::paste_sp_m_s (
            real(sp), dimension(:, :), intent(inout), allocatable mat1,
            real(sp), intent(in) sca2 )  [private]
```

16.4.2.42 paste_sp_m_v()

```
subroutine mo_append::paste_sp_m_v (
            real(sp), dimension(:, :), intent(inout), allocatable mat1,
            real(sp), dimension(:), intent(in) vec2,
            real(sp), intent(in), optional fill_value )  [private]
```

16.5 mo_canopy_interc Module Reference

Canopy interception.

Functions/Subroutines

- elemental pure subroutine, public canopy_interc (pet, interc_max, precip, interc, throughfall, evap_canopy)
  Canopy interception.

16.5.1 Detailed Description

Canopy interception.

This module deals with processes related to canopy interception, evaporation and throughfall.

Authors

Vladyslav Prykhodko

Date

Dec 2012

16.5.2 Function/Subroutine Documentation

16.5.2.1 canopy_interc()

```
elemental pure subroutine, public mo_canopy_interc::canopy_interc (
            real(dp), intent(in) pet,
            real(dp), intent(in) interc_max,
            real(dp), intent(in) precip,
            real(dp), intent(inout) interc,
            real(dp), intent(out) throughfall,
            real(dp), intent(out) evap_canopy )
```

Canopy interception.

Calculates throughfall. Updates interception and evaporation intensity from canopy. Throughfall (F) is estimated as a function of the incoming precipitation (P), the current status of the canopy water content (C), and the max. water

$$F = Max((P + C - C_{max}), 0)$$

Evaporation (E) from canopy is estimated as a fraction of the potential evapotranspiration(E_p) depending on the current status of the canopy water content (C) and the max. water content(C_{max}) that can be intecepted by the vegetation.

$$E = E_p(C/C_{max})^{2/3}$$

ADDITIONAL INFORMATION content(C_{max}) that can be intecepted by the vegetation. canopy_interc(pet, interc↩
_month_max, interc_max, precip, throughfall, evap_canopy, interc)

Parameters

in	*REAL(dp) :: pet*	Potential evapotranspiration [mm s-1]
in	*REAL(dp) :: interc_max*	Maximum interception [mm]
in	*REAL(dp) :: precip*	Daily mean precipitation [mm]
in,out	*REAL(dp) :: interc*	Interception [mm]
out	*REAL(dp) :: throughfall*	Throughfall [mm s-1]
out	*REAL(dp) :: evap_canopy*	Real evaporation intensity from canopy[mm s-1]

Authors

Vladyslav Prykhodko

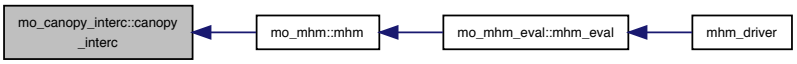
Date

Dec 2012

References mo_common_constants::eps_dp, and mo_constants::twothird_dp.

Referenced by mo_mhm::mhm().

Here is the caller graph for this function:



16.6 mo_common_constants Module Reference

Provides constants commonly used by mHM, mRM and MPR.

Variables

- real(dp), parameter, public eps_dp = epsilon(1.0_dp)

  epsilon(1.0) in double precision
- real(sp), parameter, public eps_sp = epsilon(1.0_sp)

  epsilon(1.0) in single precision
- integer(i4), parameter, public nodata_i4 = -9999_i4
- real(dp), parameter, public nodata_dp = -9999._dp
- real(dp), parameter, public p1_initstatefluxes = 0.00_dp
- integer(i4), parameter, public ncolpars = 5_i4
- integer(i4), parameter, public maxnobasins = 50_i4
- integer(i4), parameter, public maxnlcovers = 50_i4
- real(dp), parameter, public dayhours = 24.0_dp
- real(dp), parameter, public yearmonths = 12.0_dp
- integer(i4), parameter, public yearmonths_i4 = 12
- real(dp), parameter, public yeardays = 365.0_dp
- real(dp), parameter, public daysecs = 86400.0_dp
- real(dp), parameter, public hoursecs = 3600.0_dp

16.6.1 Detailed Description

Provides constants commonly used by mHM, mRM and MPR.

Provides commonly used by mHM, mRM and MPR such as no_data values and eps

Authors

Robert Schweppe

Date

Dec 2017

16.6.2 Variable Documentation

16.6.2.1 dayhours

```
real(dp), parameter, public mo_common_constants::dayhours = 24.0_dp
```

16.6.2.2 daysecs

```
real(dp), parameter, public mo_common_constants::daysecs = 86400.0_dp
```

Referenced by mo_pet::extraterr_rad_approx(), mo_pet::pet_penman(), and mo_pet::pet_priestly().

16.6.2.3 eps_dp

```
real(dp), parameter, public mo_common_constants::eps_dp = epsilon(1.0_dp)
```

epsilon(1.0) in double precision

Referenced by mo_multi_param_reg::aerodynamical_resistance(), mo_canopy_interc::canopy_interc(), mo_temporal_aggregation::day2mon_average_dp(), mo_objective_function::extract_basin_avg_tws(), mo_mpr_startup::l0_check_input(), mo_mpr_read_config::mpr_read_config(), mo_objective_function::objective_kge_q_rmse_et(), mo_objective_function::objective_kge_q_rmse_tws(), mo_soil_database::read_soil_lut(), mo_runoff::runoff_unsat_zone(), and mo_soil_moisture::soil_moisture().

16.6.2.4 eps_sp

```
real(sp), parameter, public mo_common_constants::eps_sp = epsilon(1.0_sp)
```

epsilon(1.0) in single precision

16.6.2.5 hoursecs

```
real(dp), parameter, public mo_common_constants::hoursecs = 3600.0_dp
```

Referenced by mo_mrm_routing::l11_runoff_acc(), mo_mhm_eval::mhm_eval(), mo_mrm_eval::mrm_eval(), mo_mrm_mpr::mrm_init_param(), mo_mrm_read_data::mrm_read_bankfull_runoff(), mo_mrm_read_data::mrm_read_total_runoff(), and mo_mrm_mpr::mrm_update_param().

16.6.2.6 maxnlcovers

```
integer(i4), parameter, public mo_common_constants::maxnlcovers = 50_i4
```

Referenced by mo_common_read_config::common_read_config().

16.6.2.7 maxnobasins

```
integer(i4), parameter, public mo_common_constants::maxnobasins = 50_i4
```

Referenced by mo_common_mhm_mrm_read_config::common_mhm_mrm_read_config(), mo_common_read_config::common_read_config(), mo_mhm_read_config::mhm_read_config(), mo_mpr_read_config::mpr_read_config(), and mo_mrm_read_config::mrm_read_config().

16.6.2.8 ncolpars

```
integer(i4), parameter, public mo_common_constants::ncolpars = 5_i4
```

Referenced by mo_mpr_read_config::mpr_read_config(), and mo_mrm_read_config::read_mrm_routing_params().

16.6.2.9 nodata_dp

```
real(dp), parameter, public mo_common_constants::nodata_dp = -9999._dp
```

Referenced by mo_mrm_river_head::avg_and_write_timestep(), mo_multi_param_reg::baseflow_param(), mo_meteo_forcings::chunk_config(), mo_mrm_river_head::create_output(), mo_mrm_write_fluxes_states::createoutputfile(), mo_write_fluxes_states::createoutputfile(), mo_objective_function::extract_basin_avg_tws(), mo_soil_database::generate_soil_database(), mo_grid::init_lowres_level(), mo_mrm_river_head::init_masked_zeros_l0(), mo_multi_param_reg::karstic_layer(), mo_mrm_net_startup::l11_calc_celerity(), mo_mrm_net_startup::l11_flow_accumulation(), mo_mrm_net_startup::l11_fraction_sealed_floodplain(), mo_mrm_routing::l11_runoff_acc(), mo_mrm_net_startup::l11_stream_features(), mo_meteo_forcings::meteo_forcings_wrapper(), mo_meteo_forcings::meteo_weights_wrapper(), mo_mhm_eval::mhm_eval(), mo_multi_param_reg::mpr(), mo_mpr_read_config::mpr_read_config(), mo_mpr_runoff::mpr_runoff(), mo_mpr_soilmoist::mpr_sm(), mo_mpr_smhorizons::mpr_smhorizons(), mo_mrm_init::mrm_init(), mo_mrm_read_data::mrm_read_discharge(), mo_mrm_read_data::mrm_read_l0_data(), mo_mrm_restart::mrm_read_restart_config(), mo_mrm_read_data::mrm_read_total_runoff(), mo_mrm_restart::mrm_write_restart(), mo_objective_function::objective(), mo_objective_function::objective_et_kge_catchment_avg(), mo_objective_function::objective_kge_q_rmse_et(), mo_objective_function::objective_kge_q_rmse_tws(), mo_objective_function::objective_neutrons_kge_catchment_avg(), mo_objective_function::objective_sm_kge_catchment_avg(), mo_objective_function::objective_sm_pd(), mo_read_optional_data::read_basin_avg_tws(), mo_read_wrapper::read_data(), mo_common_read_data::read_dem(), mo_read_optional_data::read_evapotranspiration(), mo_read_optional_data::read_neutrons(), mo_soil_database::read_soil_lut(), mo_read_optional_data::read_soil_moisture(), mo_spatial_agg_disagg_forcing::spatial_aggregation_3d(), mo_spatial_agg_disagg_forcing::spatial_aggregation_4d(), mo_spatial_agg_disagg_forcing::spatial_disaggregation_3d(), mo_spatial_agg_disagg_forcing::spatial_disaggregation_4d(), mo_write_ascii::write_configfile(), mo_mrm_write::write_configfile(), mo_mpr_restart::write_eff_params(), mo_common_restart::write_grid_info(), mo_restart::write_restart_files(), mo_mrm_write_fluxes_states::writevariableattributes(),

mo_write_fluxes_states::writevariableattributes(), mo_mrm_write_fluxes_states::writevariabletimestep(), and mo↩
_write_fluxes_states::writevariabletimestep().

16.6.2.10 nodata_i4

```
integer(i4), parameter, public mo_common_constants::nodata_i4 = -9999_i4
```

Referenced by mo_mrm_river_head::calc_channel_elevation(), mo_soil_database::generate_soil_database(), mo_grid::init_lowres_level(), mo_multi_param_reg::karstic_layer(), mo_mrm_init::l0_check_input_routing(), mo_↩
upscaling_operators::l0_fractionalcover_in_lx(), mo_mrm_net_startup::l11_calc_celerity(), mo_mrm_net_startup↩
::l11_flow_accumulation(), mo_mrm_net_startup::l11_flow_direction(), mo_mrm_net_startup::l11_l1_mapping(), mo_mrm_net_startup::l11_link_location(), mo_mrm_net_startup::l11_routing_order(), mo_mrm_net_startup::l11↩
_set_drain_outlet_gauges(), mo_mrm_net_startup::l11_set_network_topology(), mo_mrm_net_startup::l11_↩
stream_features(), mo_mhm_read_config::mhm_read_config(), mo_multi_param_reg::mpr(), mo_mpr_runoff↩
::mpr_runoff(), mo_mpr_soilmoist::mpr_sm(), mo_mrm_init::mrm_init(), mo_mrm_read_config::mrm_read_config(), mo_mrm_read_data::mrm_read_l0_data(), mo_mrm_restart::mrm_write_restart(), mo_read_wrapper::read_↩
data(), mo_common_read_data::read_lcover(), mo_soil_database::read_soil_lut(), mo_mrm_read_data::rotate_↩
fdir_variable(), mo_common_read_config::set_land_cover_scenes_id(), mo_mpr_restart::write_eff_params(), and mo_common_restart::write_grid_info().

16.6.2.11 p1_initstatefluxes

```
real(dp), parameter, public mo_common_constants::p1_initstatefluxes = 0.00_dp
```

Referenced by mo_mpr_startup::init_eff_params(), mo_init_states::variables_alloc(), mo_init_states::variables_↩
default_init(), and mo_mrm_init::variables_default_init_routing().

16.6.2.12 yeardays

```
real(dp), parameter, public mo_common_constants::yeardays = 365.0_dp
```

Referenced by mo_pet::extraterr_rad_approx().

16.6.2.13 yearmonths

```
real(dp), parameter, public mo_common_constants::yearmonths = 12.0_dp
```

Referenced by mo_read_lut::read_lai_lut().

16.6.2.14 yearmonths_i4

```
integer(i4), parameter, public mo_common_constants::yearmonths_i4 = 12
```

Referenced by mo_mpr_startup::init_eff_params(), and mo_read_wrapper::read_data().

16.7 mo_common_file Module Reference

Provides file names and units for mRM.

Variables

- character(len= ∗), parameter file_dem = 'dem.asc'
  DEM input data file.
- integer, parameter udem = 53
  Unit for DEM input data file.
- integer, parameter ulcoverclass = 61
  Unit for LCover input data file.
- character(len= ∗), parameter file_config = 'ConfigFile.log'
  file defining mHM's outputs
- integer, parameter uconfig = 68
  Unit for file defining mHM's outputs.

16.7.1 Detailed Description

Provides file names and units for mRM.

Provides all filenames as well as all units used for the multiscale Routing Model mRM.

Authors

Matthias Cuntz, Stephan Thober

Date

Aug 2015

16.7.2 Variable Documentation

16.7.2.1 file_config

```
character(len=∗), parameter mo_common_file::file_config = 'ConfigFile.log'
```

file defining mHM's outputs

Referenced by mo_write_ascii::write_configfile(), and mo_mrm_write::write_configfile().

16.7.2.2 file_dem

```
character(len=∗), parameter mo_common_file::file_dem = 'dem.asc'
```

DEM input data file.

Referenced by mo_common_read_data::read_dem().

16.7.2.3 uconfig

```
integer, parameter mo_common_file::uconfig = 68
```

Unit for file defining mHM's outputs.

Referenced by mo_write_ascii::write_configfile(), and mo_mrm_write::write_configfile().

16.7.2.4 udem

```
integer, parameter mo_common_file::udem = 53
```

Unit for DEM input data file.

Referenced by mo_common_read_data::read_dem().

16.7.2.5 ulcoverclass

```
integer, parameter mo_common_file::ulcoverclass = 61
```

Unit for LCover input data file.

Referenced by mo_common_read_data::read_lcover().

16.8 mo_common_functions Module Reference

Provides small utility functions used by multiple parts of the code (mHM, mRM, MPR)

Functions/Subroutines

- logical function, public in_bound (params)
  TODO: add description.

16.8.1 Detailed Description

Provides small utility functions used by multiple parts of the code (mHM, mRM, MPR)

Provides the functions in_bound used to check global_parameter ranges

Authors

Robert Schweppe

Date

Dec 2017

16.8.2 Function/Subroutine Documentation

16.8.2.1 in_bound()

```
logical function, public mo_common_functions::in_bound (
            real(dp), dimension(:, :), intent(in) params )
```

TODO: add description.

TODO: add description

Parameters

in	*real(dp), dimension(:, :) :: params*	parameter: col_1=Lower bound, col_2=Upper bound col_3=initial

Authors

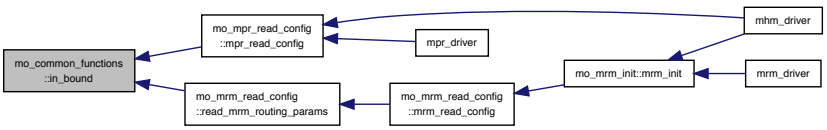
Robert Schweppe

Date

Jun 2018

Referenced by mo_mpr_read_config::mpr_read_config(), and mo_mrm_read_config::read_mrm_routing_params().

Here is the caller graph for this function:



16.9 mo_common_mhm_mrm_file Module Reference

Provides file names and units for mHM.

Variables

- character(len=∗), parameter file_opti = 'FinalParam.out'

  file defining optimization outputs (objective and parameter set)
- integer, parameter uopti = 72

  Unit for file optimization outputs (objective and parameter set)
- character(len=∗), parameter file_opti_nml = 'FinalParam.nml'

  file defining optimization outputs in a namelist format (parameter set)
- integer, parameter uopti_nml = 73

  Unit for file optimization outputs in a namelist format (parameter set)

16.9.1 Detailed Description

Provides file names and units for mHM.

Provides all filenames as well as all units used for the hydrologic model mHM.

Authors

Matthias Cuntz

Date

Jan 2012

16.9.2 Variable Documentation

16.9.2.1 file_opti

```
character(len = *), parameter mo_common_mhm_mrm_file::file_opti = 'FinalParam.out'
```

file defining optimization outputs (objective and parameter set)

Referenced by mo_mrm_write::mrm_write_optifile(), and mo_write_ascii::write_optifile().

16.9.2.2 file_opti_nml

```
character(len = *), parameter mo_common_mhm_mrm_file::file_opti_nml = 'FinalParam.nml'
```

file defining optimization outputs in a namelist format (parameter set)

Referenced by mo_mrm_write::mrm_write_optinamelist(), and mo_write_ascii::write_optinamelist().

16.9.2.3 uopti

```
integer, parameter mo_common_mhm_mrm_file::uopti = 72
```

Unit for file optimization outputs (objective and parameter set)

Referenced by mo_mrm_write::mrm_write_optifile(), and mo_write_ascii::write_optifile().

16.9.2.4 uopti_nml

```
integer, parameter mo_common_mhm_mrm_file::uopti_nml = 73
```

Unit for file optimization outputs in a namelist format (parameter set)

Referenced by mo_mrm_write::mrm_write_optinamelist(), and mo_write_ascii::write_optinamelist().

16.10 mo_common_mhm_mrm_read_config Module Reference

Reading of main model configurations.

Functions/Subroutines

- subroutine, public common_mhm_mrm_read_config (file_namelist, unamelist)

  Read main configurations for common parts.
- subroutine, public check_optimization_settings

  TODO: add description.
- subroutine, public common_check_resolution (do_message, allow_subgrid_routing)

  TODO: add description.

16.10.1 Detailed Description

Reading of main model configurations.

This routine reads the configurations of common program parts

Authors

Matthias Zink

Date

Dec 2012

16.10.2 Function/Subroutine Documentation

16.10.2.1 check_optimization_settings()

```
subroutine, public mo_common_mhm_mrm_read_config::check_optimization_settings ( )
```

TODO: add description.

TODO: add description

Authors

Robert Schweppe

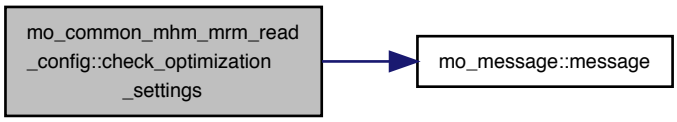
Date

Jun 2018

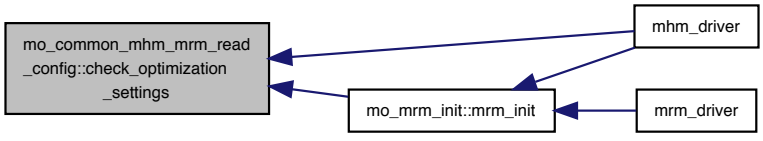
References mo_common_mhm_mrm_variables::dds_r, mo_common_variables::global_parameters, mo_message::message(), mo_common_mhm_mrm_variables::niterations, mo_common_mhm_mrm_variables::sce_ngs, mo_common_mhm_mrm_variables::sce_npg, and mo_common_mhm_mrm_variables::sce_nps.

Referenced by mhm_driver(), and mo_mrm_init::mrm_init().

Here is the call graph for this function:



Here is the caller graph for this function:



16.10.2.2 common_check_resolution()

```
subroutine, public mo_common_mhm_mrm_read_config::common_check_resolution (
            logical, intent(in) do_message,
            logical, intent(in) allow_subgrid_routing )
```

TODO: add description.

TODO: add description

Parameters

in	*logical :: do_message*	
in	*logical :: allow_subgrid_routing*	

Authors

Robert Schweppe

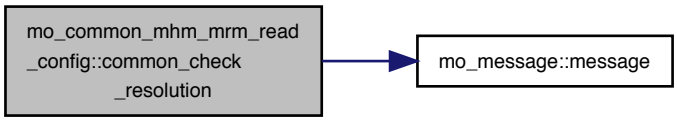
Date

Jun 2018

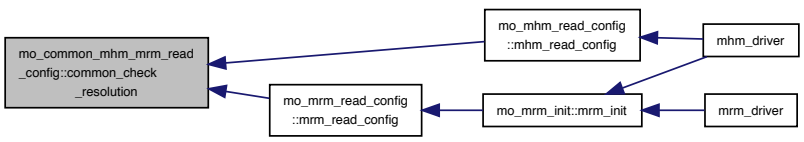
References mo_message::message(), mo_common_variables::nbasins, mo_common_variables::resolutionhydrology, and mo_common_mhm_mrm_variables::resolutionrouting.

Referenced by mo_mhm_read_config::mhm_read_config(), and mo_mrm_read_config::mrm_read_config().

Here is the call graph for this function:



Here is the caller graph for this function:



16.10.2.3 common_mhm_mrm_read_config()

```
subroutine, public mo_common_mhm_mrm_read_config::common_mhm_mrm_read_config (
             character(*), intent(in) file_namelist,
             integer, intent(in) unamelist )
```

Read main configurations for common parts.

TODO: add description

Parameters

in	*character(*) :: file_namelist*	
in	*integer :: unamelist*	

Authors

Matthias Zink

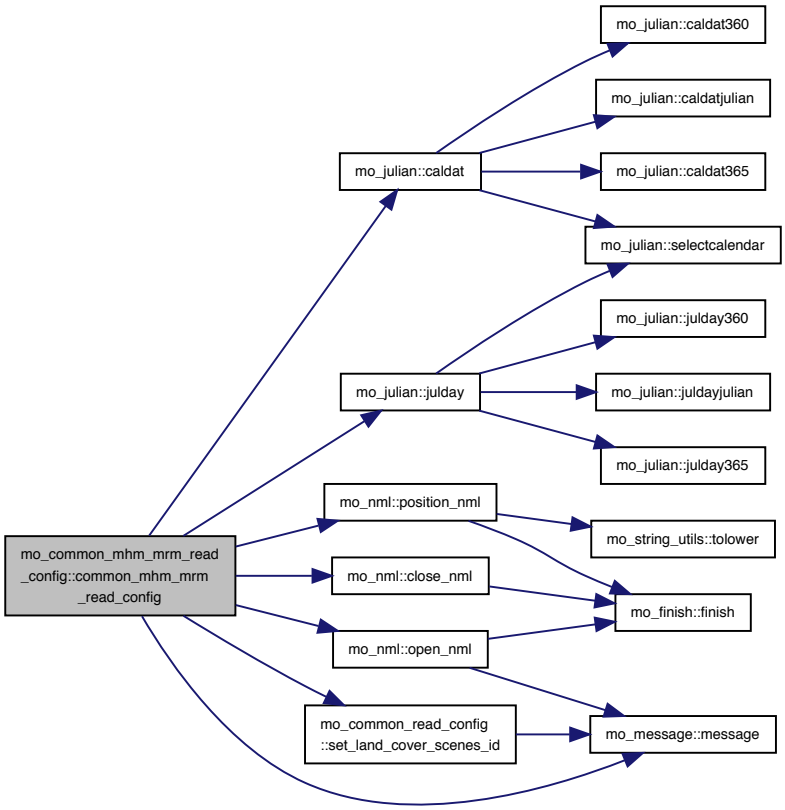
Date

Dec 2012

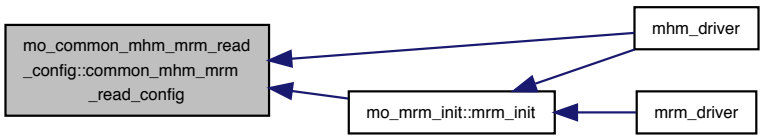
References mo_julian::caldat(), mo_nml::close_nml(), mo_common_mhm_mrm_variables::dds_r, mo_common_↩ mhm_mrm_variables::dirrestartin, mo_common_mhm_mrm_variables::evalper, mo_julian::julday(), mo_common↩ _variables::lcfilename, mo_common_mhm_mrm_variables::lcyearid, mo_common_constants::maxnobasins, mo↩ _common_mhm_mrm_variables::mcmc_error_params, mo_common_mhm_mrm_variables::mcmc_opti, mo_↩ message::message(), mo_common_variables::nbasins, mo_common_mhm_mrm_variables::niterations, mo↩ _common_mhm_mrm_variables::ntstepday, mo_nml::open_nml(), mo_common_mhm_mrm_variables::opti_↩ function, mo_common_mhm_mrm_variables::opti_method, mo_common_mhm_mrm_variables::optimize, mo_↩ common_mhm_mrm_variables::optimize_restart, mo_nml::position_nml(), mo_common_mhm_mrm_variables↩ ::read_restart, mo_common_mhm_mrm_variables::resolutionrouting, mo_common_mhm_mrm_variables::sa_↩ temp, mo_common_mhm_mrm_variables::sce_ngs, mo_common_mhm_mrm_variables::sce_npg, mo_common↩ _mhm_mrm_variables::sce_nps, mo_common_mhm_mrm_variables::seed, mo_common_read_config::set_land↩ _cover_scenes_id(), mo_common_mhm_mrm_variables::simper, mo_common_mhm_mrm_variables::timestep, mo_common_mhm_mrm_variables::warmingdays, and mo_common_mhm_mrm_variables::warmper.

Referenced by mhm_driver(), and mo_mrm_init::mrm_init().

Here is the call graph for this function:



Here is the caller graph for this function:



16.11 mo_common_mhm_mrm_variables Module Reference

Provides structures needed by mHM, mRM and/or mpr.

Variables

- integer(i4) mrm_coupling_mode
- integer(i4), public timestep

- real(dp), public c2tstu
- real(dp), dimension(:), allocatable, public resolutionrouting
- logical, public read_restart
- type(period), dimension(:), allocatable, public warmper
- type(period), dimension(:), allocatable, public evalper
- type(period), dimension(:), allocatable, public simper
- type(period), public readper
- integer(i4), dimension(:), allocatable, public warmingdays
- integer(i4), dimension(:, :), allocatable, public lcyearid
- integer(i4), public ntstepday
- character(256), dimension(:), allocatable, public dirrestartin
- integer(i4), public opti_method
- integer(i4), public opti_function
- logical, public optimize
- logical, public optimize_restart
- integer(i8), public seed
- integer(i4), public niterations
- real(dp), public dds_r
- real(dp), public sa_temp
- integer(i4), public sce_ngs
- integer(i4), public sce_npg
- integer(i4), public sce_nps
- logical, public mcmc_opti
- integer(i4), parameter, public nerror_model = 2
- real(dp), dimension(nerror_model), public mcmc_error_params

16.11.1 Detailed Description

Provides structures needed by mHM, mRM and/or mpr.

Provides the global structure period that is used by both mHM and mRM.

Authors

Stephan Thober

Date

Sep 2015

16.11.2 Variable Documentation

16.11.2.1 c2tstu

```
real(dp), public mo_common_mhm_mrm_variables::c2tstu
```

Referenced by mo_startup::constants_init(), and mo_mhm_eval::mhm_eval().

16.11.2.2 dds_r

```
real(dp), public mo_common_mhm_mrm_variables::dds_r
```

Referenced by mo_common_mhm_mrm_read_config::check_optimization_settings(), mo_common_mhm_mrm_↩ read_config::common_mhm_mrm_read_config(), and mo_optimization::optimization().

16.11.2.3 dirrestartin

```
character(256), dimension(:), allocatable, public mo_common_mhm_mrm_variables::dirrestartin
```

Referenced by mo_common_mhm_mrm_read_config::common_mhm_mrm_read_config(), mo_mhm_eval::mhm↩ _eval(), mo_startup::mhm_initialize(), mo_mrm_eval::mrm_eval(), and mo_mrm_init::mrm_init().

16.11.2.4 evalper

```
type(period), dimension(:), allocatable, public mo_common_mhm_mrm_variables::evalper
```

Referenced by mo_common_mhm_mrm_read_config::common_mhm_mrm_read_config(), mo_mrm_river_↩ head::create_output(), mo_mrm_write_fluxes_states::createoutputfile(), mo_write_fluxes_states::createoutputfile(), mo_objective_function::extract_basin_avg_tws(), mo_mrm_objective_function_runoff::extract_runoff(), mo_mrm↩ _read_data::mrm_read_discharge(), mo_mrm_write::mrm_write(), mo_mrm_objective_function_runoff::multi↩ _objective_ae_fdc_lsv_nse_djf(), mo_objective_function::objective_kge_q_rmse_et(), mo_objective_function↩ ::objective_kge_q_rmse_tws(), mo_read_optional_data::read_basin_avg_tws(), mo_read_optional_data::read↩ _evapotranspiration(), mo_read_optional_data::read_neutrons(), mo_read_optional_data::read_soil_moisture(), mo_write_ascii::write_configfile(), mo_mrm_write::write_configfile(), and mo_mrm_write::write_daily_obs_sim_↩ discharge().

16.11.2.5 lcyearid

```
integer(i4), dimension(:, :), allocatable, public mo_common_mhm_mrm_variables::lcyearid
```

Referenced by mo_common_mhm_mrm_read_config::common_mhm_mrm_read_config(), mo_mhm_eval::mhm↩ _eval(), mo_mrm_eval::mrm_eval(), mo_write_ascii::write_configfile(), and mo_mrm_write::write_configfile().

16.11.2.6 mcmc_error_params

```
real(dp), dimension(nerror_model), public mo_common_mhm_mrm_variables::mcmc_error_params
```

Referenced by mo_common_mhm_mrm_read_config::common_mhm_mrm_read_config(), and mo_optimization↩ ::optimization().

16.11.2.7 mcmc_opti

```
logical, public mo_common_mhm_mrm_variables::mcmc_opti
```

Referenced by mo_common_mhm_mrm_read_config::common_mhm_mrm_read_config(), and mo_optimization↩ ::optimization().

16.11.2.8 mrm_coupling_mode

```
integer(i4) mo_common_mhm_mrm_variables::mrm_coupling_mode
```

Referenced by mrm_driver(), mo_mrm_init::mrm_init(), mo_mrm_write::mrm_write(), mo_mrm_restart::mrm_↩ write_restart(), and mo_mrm_write::write_configfile().

16.11.2.9 nerror_model

```
integer(i4), parameter, public mo_common_mhm_mrm_variables::nerror_model = 2
```

16.11.2.10 niterations

```
integer(i4), public mo_common_mhm_mrm_variables::niterations
```

Referenced by mo_common_mhm_mrm_read_config::check_optimization_settings(), mo_common_mhm_mrm_↩ read_config::common_mhm_mrm_read_config(), and mo_optimization::optimization().

16.11.2.11 ntstepday

```
integer(i4), public mo_common_mhm_mrm_variables::ntstepday
```

Referenced by mo_meteo_forcings::chunk_size(), mo_common_mhm_mrm_read_config::common_mhm_mrm↩ _read_config(), mo_objective_function::extract_basin_avg_tws(), mo_mrm_objective_function_runoff::extract_↩ runoff(), mo_write_fluxes_states::fluxesunit(), mo_meteo_forcings::is_read(), mhm_driver(), mo_mhm_eval::mhm↩ _eval(), mo_mrm_eval::mrm_eval(), mo_mrm_read_data::mrm_read_discharge(), mo_mrm_write::mrm_write(), and mo_read_optional_data::read_basin_avg_tws().

16.11.2.12 opti_function

```
integer(i4), public mo_common_mhm_mrm_variables::opti_function
```

Referenced by mo_common_mhm_mrm_read_config::common_mhm_mrm_read_config(), mo_mhm_read_↩ config::mhm_read_config(), mo_mrm_read_data::mrm_read_discharge(), mo_mrm_objective_function_runoff↩ ::multi_objective_runoff(), mo_objective_function::objective(), mo_optimization::optimization(), mo_read_optional↩ _data::read_basin_avg_tws(), and mo_mrm_objective_function_runoff::single_objective_runoff().

16.11.2.13 opti_method

```
integer(i4), public mo_common_mhm_mrm_variables::opti_method
```

Referenced by mo_common_mhm_mrm_read_config::common_mhm_mrm_read_config(), mo_optimization↩ ::optimization(), and mo_mrm_objective_function_runoff::single_objective_runoff().

16.11.2.14 optimize

```
logical, public mo_common_mhm_mrm_variables::optimize
```

Referenced by mo_common_mhm_mrm_read_config::common_mhm_mrm_read_config(), mo_mhm_eval::mhm↩
_eval(), mo_mhm_read_config::mhm_read_config(), mrm_driver(), mo_mrm_eval::mrm_eval(), mo_mrm_mpr↩
::mrm_init_param(), mo_mrm_read_data::mrm_read_discharge(), mo_mrm_mpr::mrm_update_param(), and mo↩
_read_optional_data::read_basin_avg_tws().

16.11.2.15 optimize_restart

```
logical, public mo_common_mhm_mrm_variables::optimize_restart
```

Referenced by mo_common_mhm_mrm_read_config::common_mhm_mrm_read_config(), and mo_optimization↩
::optimization().

16.11.2.16 read_restart

```
logical, public mo_common_mhm_mrm_variables::read_restart
```

Referenced by mo_common_mhm_mrm_read_config::common_mhm_mrm_read_config(), mo_mhm_eval::mhm↩
_eval(), mo_startup::mhm_initialize(), mo_mrm_eval::mrm_eval(), mo_mrm_init::mrm_init(), mo_write_ascii::write↩
_configfile(), and mo_mrm_write::write_configfile().

16.11.2.17 readper

```
type(period), public mo_common_mhm_mrm_variables::readper
```

Referenced by mo_meteo_forcings::meteo_forcings_wrapper(), mo_mhm_eval::mhm_eval(), and mo_meteo_↩
forcings::prepare_meteo_forcings_data().

16.11.2.18 resolutionrouting

```
real(dp), dimension(:), allocatable, public mo_common_mhm_mrm_variables::resolutionrouting
```

Referenced by mo_common_mhm_mrm_read_config::common_check_resolution(), mo_common_mhm_mrm_↩
read_config::common_mhm_mrm_read_config(), mo_mhm_eval::mhm_eval(), mo_mrm_eval::mrm_eval(), mo↩
_mrm_init::mrm_init(), mo_mrm_mpr::mrm_init_param(), mo_mrm_mpr::mrm_update_param(), mo_write_ascii↩
::write_configfile(), and mo_mrm_write::write_configfile().

16.11.2.19 sa_temp

```
real(dp), public mo_common_mhm_mrm_variables::sa_temp
```

Referenced by mo_common_mhm_mrm_read_config::common_mhm_mrm_read_config(), and mo_optimization↩
::optimization().

16.11.2.20 sce_ngs

```
integer(i4), public mo_common_mhm_mrm_variables::sce_ngs
```

Referenced by mo_common_mhm_mrm_read_config::check_optimization_settings(), mo_common_mhm_mrm_↩
read_config::common_mhm_mrm_read_config(), and mo_optimization::optimization().

16.11.2.21 sce_npg

```
integer(i4), public mo_common_mhm_mrm_variables::sce_npg
```

Referenced by mo_common_mhm_mrm_read_config::check_optimization_settings(), mo_common_mhm_mrm_↩
read_config::common_mhm_mrm_read_config(), and mo_optimization::optimization().

16.11.2.22 sce_nps

```
integer(i4), public mo_common_mhm_mrm_variables::sce_nps
```

Referenced by mo_common_mhm_mrm_read_config::check_optimization_settings(), mo_common_mhm_mrm_↩
read_config::common_mhm_mrm_read_config(), and mo_optimization::optimization().

16.11.2.23 seed

```
integer(i8), public mo_common_mhm_mrm_variables::seed
```

Referenced by mo_common_mhm_mrm_read_config::common_mhm_mrm_read_config(), and mo_optimization↩
::optimization().

16.11.2.24 simper

```
type(period), dimension(:), allocatable, public mo_common_mhm_mrm_variables::simper
```

Referenced by mo_meteo_forcings::chunk_size(), mo_common_mhm_mrm_read_config::common_mhm_mrm↩
_read_config(), mo_write_fluxes_states::fluxesunit(), mo_meteo_forcings::is_read(), mo_mhm_eval::mhm_eval(),
mo_mrm_eval::mrm_eval(), mo_mrm_read_data::mrm_read_discharge(), mo_mrm_read_data::mrm_read_total↩
_runoff(), mo_mrm_write::mrm_write(), mo_read_optional_data::read_basin_avg_tws(), mo_write_ascii::write_↩
configfile(), and mo_mrm_write::write_configfile().

16.11.2.25 timestep

```
integer(i4), public mo_common_mhm_mrm_variables::timestep
```

Referenced by mo_common_mhm_mrm_read_config::common_mhm_mrm_read_config(), mo_startup↩
::constants_init(), mo_write_fluxes_states::fluxesunit(), mo_meteo_forcings::is_read(), mo_mhm_eval::mhm_eval(),
mo_mrm_eval::mrm_eval(), mo_mrm_mpr::mrm_init_param(), mo_mrm_read_data::mrm_read_bankfull_runoff(),
mo_mrm_read_data::mrm_read_total_runoff(), mo_mrm_mpr::mrm_update_param(), mo_write_ascii::write_↩
configfile(), and mo_mrm_write::write_configfile().

16.11.2.26 warmingdays

```
integer(i4), dimension(:), allocatable, public mo_common_mhm_mrm_variables::warmingdays
```

Referenced by mo_common_mhm_mrm_read_config::common_mhm_mrm_read_config(), mo_objective_function::extract_basin_avg_tws(), mo_mrm_objective_function_runoff::extract_runoff(), mo_mhm_eval::mhm_eval(), mo_mrm_eval::mrm_eval(), and mo_mrm_write::mrm_write().

16.11.2.27 warmper

```
type(period), dimension(:), allocatable, public mo_common_mhm_mrm_variables::warmper
```

Referenced by mo_common_mhm_mrm_read_config::common_mhm_mrm_read_config(), mo_write_ascii::write_configfile(), and mo_mrm_write::write_configfile().

16.12 mo_common_read_config Module Reference

Reading of main model configurations.

Functions/Subroutines

- subroutine, public common_read_config (file_namelist, unamelist)

  Read main configurations commonly used by mHM, mRM and MPR.
- subroutine, public set_land_cover_scenes_id (sim_Per, LCyear_Id, LCfilename)

  Read main configurations commonly used by mHM, mRM and MPR.

16.12.1 Detailed Description

Reading of main model configurations.

This routine reads the configurations of namelists commonly used by mHM, mRM and MPR

Authors

Matthias Zink

Date

Dec 2012

16.12.2 Function/Subroutine Documentation

16.12.2.1 common_read_config()

```
subroutine, public mo_common_read_config::common_read_config (
             character(*), intent(in) file_namelist,
             integer, intent(in) unamelist )
```

Read main configurations commonly used by mHM, mRM and MPR.

Read the main configurations commonly used by mHM, mRM and MPR, namely: project_description, directories_general, mainconfig, processSelection, LCover

Parameters

in	*character(∗) :: file_namelist*	name of file
in	*integer :: unamelist*	id of file

Authors

Matthias Zink

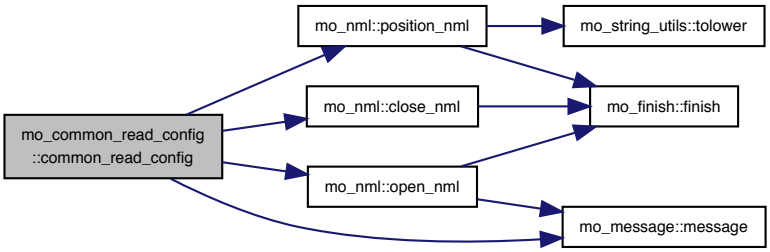
Date

Dec 2012

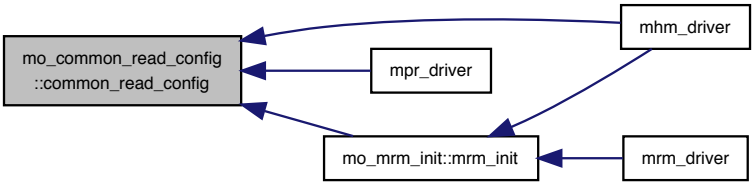
References mo_nml::close_nml(), mo_common_variables::contact, mo_common_variables::conventions, mo_common_variables::dircommonfiles, mo_common_variables::dirconfigout, mo_common_variables::dirlcover, mo_common_variables::dirmorpho, mo_common_variables::dirout, mo_common_variables::dirrestartout, mo_common_variables::filelatlon, mo_common_variables::history, mo_common_variables::iflag_cordinate_sys, mo_common_variables::l0_basin, mo_common_variables::lc_year_end, mo_common_variables::lc_year_start, mo_common_variables::lcfilename, mo_common_constants::maxnlcovers, mo_common_constants::maxnobasins, mo_message::message(), mo_common_variables::mhm_details, mo_common_variables::nbasins, mo_common_variables::nlcoverscene, mo_common_variables::nprocesses, mo_common_variables::nuniquel0basins, mo_nml::open_nml(), mo_nml::position_nml(), mo_common_variables::processmatrix, mo_common_variables::project_details, mo_common_variables::resolutionhydrology, mo_common_variables::setup_description, mo_common_variables::simulation_type, and mo_common_variables::write_restart.

Referenced by mhm_driver(), mpr_driver(), and mo_mrm_init::mrm_init().

Here is the call graph for this function:



Here is the caller graph for this function:



16.12.2.2 set_land_cover_scenes_id()

```
subroutine, public mo_common_read_config::set_land_cover_scenes_id (
            type(period), dimension(:), intent(in) sim_Per,
            integer(i4), dimension(:, :), intent(inout), allocatable LCyear_Id,
            character(256), dimension(:), intent(inout), allocatable LCfilename )
```

Read main configurations commonly used by mHM, mRM and MPR.

Read the main configurations commonly used by mHM, mRM and MPR, namely: project_description, directories↩ _general, mainconfig, processSelection, LCover

Parameters

in	*type(period), dimension(:) :: sim_Per*	
in,out	*integer(i4), dimension(:, :) :: LCyear_Id*	
in,out	*character(256), dimension(:) :: LCfilename*	

Authors

Matthias Zink

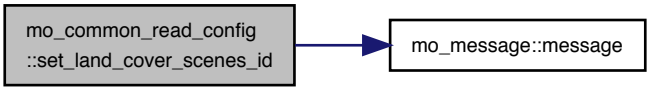
Date

Dec 2012

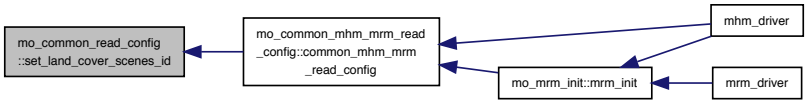
References mo_common_variables::lc_year_end, mo_common_variables::lc_year_start, mo_message↩ ::message(), mo_common_variables::nbasins, mo_common_variables::nlcoverscene, and mo_common_↩ constants::nodata_i4.

Referenced by mo_common_mhm_mrm_read_config::common_mhm_mrm_read_config().

Here is the call graph for this function:



Here is the caller graph for this function:



16.13 mo_common_read_data Module Reference

TODO: add description.

Functions/Subroutines

- subroutine, public read_dem

  TODO: add description.
- subroutine, public read_lcover

  TODO: add description.

16.13.1 Detailed Description

TODO: add description.

TODO: add description

Authors

Robert Schweppe

Date

Jun 2018

16.13.2 Function/Subroutine Documentation

16.13.2.1 read_dem()

```
subroutine, public mo_common_read_data::read_dem ( )
```

TODO: add description.

TODO: add description

Authors

Robert Schweppe

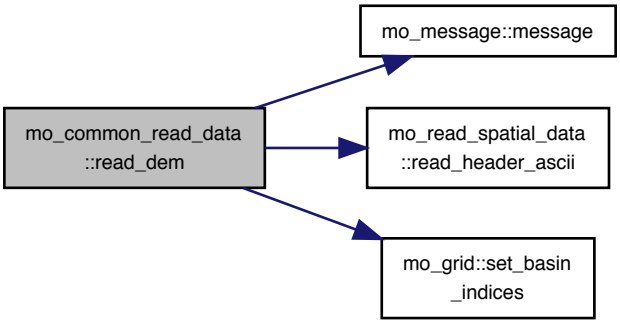
Date

Jun 2018

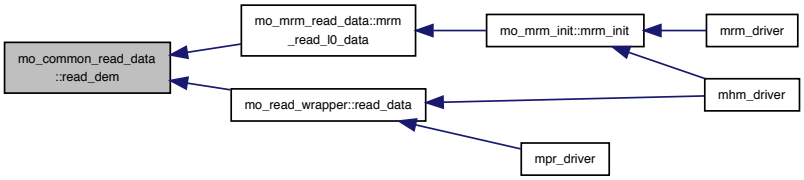
References mo_common_variables::dirmorpho, mo_common_file::file_dem, mo_common_variables::l0_basin, mo_common_variables::l0_elev, mo_common_variables::level0, mo_message::message(), mo_common_↩variables::nbasins, mo_common_constants::nodata_dp, mo_common_variables::nuniquel0basins, mo_read_↩spatial_data::read_header_ascii(), mo_common_variables::resolutionhydrology, mo_grid::set_basin_indices(), and mo_common_file::udem.

Referenced by mo_mrm_read_data::mrm_read_l0_data(), and mo_read_wrapper::read_data().

Here is the call graph for this function:



Here is the caller graph for this function:



16.13.2.2 read_lcover()

```
subroutine, public mo_common_read_data::read_lcover ( )
```

TODO: add description.

TODO: add description

Authors

Robert Schweppe

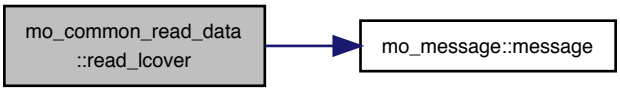
Date

Jun 2018

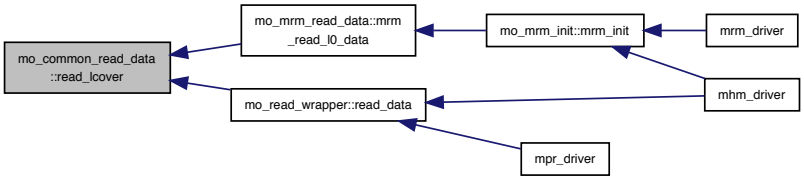
References mo_common_variables::dirlcover, mo_common_variables::l0_basin, mo_common_variables::l0_lcover, mo_common_variables::lcfilename, mo_common_variables::level0, mo_message::message(), mo_common_↩ variables::nbasins, mo_common_variables::nlcoverscene, mo_common_constants::nodata_i4, and mo_common↩ _file::ulcoverclass.

Referenced by mo_mrm_read_data::mrm_read_l0_data(), and mo_read_wrapper::read_data().

Here is the call graph for this function:



Here is the caller graph for this function:



16.14 mo_common_restart Module Reference

TODO: add description.

Functions/Subroutines

- subroutine, public write_grid_info (grid_in, level_name, nc)

  write restart files for each basin
- subroutine, public read_grid_info (iBasin, InPath, level_name, fname_part, new_grid)

  reads configuration apart from Level 11 configuration from a restart directory

16.14.1 Detailed Description

TODO: add description.

TODO: add description

Authors

Robert Schweppe

Date

Jun 2018

16.14.2 Function/Subroutine Documentation

16.14.2.1 read_grid_info()

```
subroutine, public mo_common_restart::read_grid_info (
             integer(i4), intent(in) iBasin,
             character(256), intent(in) InPath,
             character(*), intent(in) level_name,
             character(*), intent(in) fname_part,
             type(grid), intent(inout) new_grid )
```

reads configuration apart from Level 11 configuration from a restart directory

read configuration variables from a given restart directory and initializes all configuration variables, that are initialized in the subroutine initialise, contained in module mo_startup.

Parameters

in	*integer(i4) :: iBasin*	number of basin
in	*character(256) :: InPath*	Input Path including trailing slash
in	*character(∗) :: level_name*	level_name (id)
in	*character(∗) :: fname_part*	filename part (either "mHM" or "mRM")
in,out	*type(Grid) :: new_grid*	grid to save information to

Authors

Stephan Thober

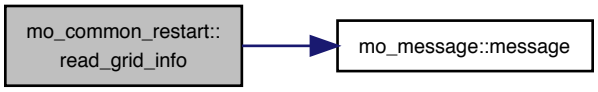
Date

Apr 2013

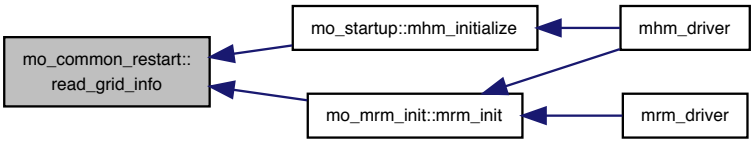
References mo_kind::dp, mo_kind::i4, and mo_message::message().

Referenced by mo_startup::mhm_initialize(), and mo_mrm_init::mrm_init().

Here is the call graph for this function:



Here is the caller graph for this function:



16.14.2.2 write_grid_info()

```
subroutine, public mo_common_restart::write_grid_info (
             type(grid), intent(in) grid_in,
             character(*), intent(in) level_name,
             type(ncdataset), intent(inout) nc )
```

write restart files for each basin

write restart files for each basin. For each basin three restart files are written. These are xxx_states.nc, xxx_L11↩ _config.nc, and xxx_config.nc (xxx being the three digit basin index). If a variable is added here, it should also be added in the read restart routines below.

Parameters

in	*type(Grid) :: grid_in*	level to be written
in	*character(∗) :: level_name*	level_id
in,out	*type(NcDataset) :: nc*	NcDataset to write information to

Authors

Stephan Thober

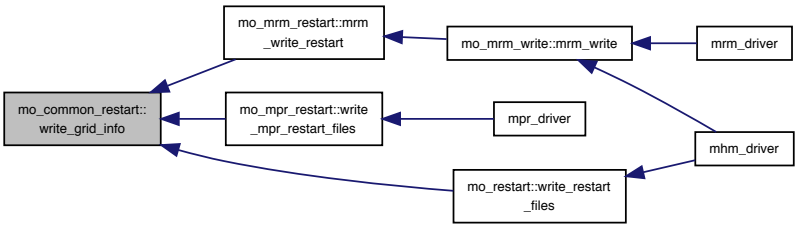
Date

Jun 2014

References mo_kind::dp, mo_kind::i4, mo_common_constants::nodata_dp, and mo_common_constants::nodata_i4.

Referenced by mo_mrm_restart::mrm_write_restart(), mo_mpr_restart::write_mpr_restart_files(), and mo_restart::write_restart_files().

Here is the caller graph for this function:



16.15 mo_common_variables Module Reference

Provides structures needed by mHM, mRM and/or mpr.

Data Types

- type grid
- type gridremapper
- type period

Variables

- character(1024), public project_details
- character(1024), public setup_description
- character(1024), public simulation_type
- character(256), public conventions
- character(1024), public contact
- character(1024), public mhm_details
- character(1024), public history
- integer(i4), public iflag_cordinate_sys
- real(dp), dimension(:), allocatable, public resolutionhydrology
- integer(i4), dimension(:), allocatable, public l0_basin
- integer(i4), dimension(:), allocatable, public l11_basin
- logical, public write_restart
- character(256), dimension(:), allocatable, public dirrestartout
- character(256), public dirconfigout
- character(256), public dircommonfiles
- character(256), dimension(:), allocatable, public dirmorpho
- character(256), dimension(:), allocatable, public dirlcover
- character(256), dimension(:), allocatable, public dirout

- character(256), dimension(:), allocatable, public filelatlon
- type(grid), dimension(:), allocatable, target, public level0
- type(grid), dimension(:), allocatable, target, public level1
- type(gridremapper), dimension(:), allocatable, public l0_l1_remap
- type(gridremapper), dimension(:), allocatable, public l0_l11_remap
- type(gridremapper), dimension(:), allocatable, public l1_l11_remap
- real(dp), dimension(:), allocatable, public l0_elev
- integer(i4), dimension(:, :), allocatable, public l0_lcover
- integer(i4), public nbasins
- integer(i4), public nuniquel0basins
- integer(i4), public nlcoverscene
- character(256), dimension(:), allocatable, public lcfilename
- integer(i4), dimension(:), allocatable, public lc_year_start
- integer(i4), dimension(:), allocatable, public lc_year_end
- integer(i4), parameter, public nprocesses = 10
- integer(i4), dimension(nprocesses, 3), public processmatrix
- real(dp), dimension(:, :), allocatable, target, public global_parameters
- character(256), dimension(:), allocatable, public global_parameters_name
- logical alma_convention

16.15.1 Detailed Description

Provides structures needed by mHM, mRM and/or mpr.

Provides the global structure period that is used by both mHM and mRM.

Authors

Stephan Thober

Date

Sep 2015

16.15.2 Variable Documentation

16.15.2.1 alma_convention

```
logical mo_common_variables::alma_convention
```

Referenced by mo_mrm_read_data::mrm_read_bankfull_runoff(), mo_mrm_read_config::mrm_read_config(), and mo_mrm_read_data::mrm_read_total_runoff().

16.15.2.2 contact

```
character(1024), public mo_common_variables::contact
```

Referenced by mo_common_read_config::common_read_config(), and mo_mrm_river_head::create_output().

16.15.2.3 conventions

```
character(256), public mo_common_variables::conventions
```

Referenced by mo_common_read_config::common_read_config(), and mo_mrm_river_head::create_output().

16.15.2.4 dircommonfiles

```
character(256), public mo_common_variables::dircommonfiles
```

Referenced by mo_common_read_config::common_read_config(), and mo_read_wrapper::read_data().

16.15.2.5 dirconfigout

```
character(256), public mo_common_variables::dirconfigout
```

Referenced by mo_common_read_config::common_read_config(), mrm_driver(), mo_mrm_write::mrm_write_optifile(), mo_mrm_write::mrm_write_optinamelist(), mo_write_ascii::write_configfile(), mo_mrm_write::write_configfile(), mo_write_ascii::write_optifile(), and mo_write_ascii::write_optinamelist().

16.15.2.6 dirlcover

```
character(256), dimension(:), allocatable, public mo_common_variables::dirlcover
```

Referenced by mo_common_read_config::common_read_config(), mo_mrm_init::config_output(), mo_common_read_data::read_lcover(), mo_write_ascii::write_configfile(), and mo_mrm_write::write_configfile().

16.15.2.7 dirmorpho

```
character(256), dimension(:), allocatable, public mo_common_variables::dirmorpho
```

Referenced by mo_common_read_config::common_read_config(), mo_mrm_init::config_output(), mo_mrm_read_data::mrm_read_l0_data(), mo_read_wrapper::read_data(), mo_common_read_data::read_dem(), mo_write_ascii::write_configfile(), and mo_mrm_write::write_configfile().

16.15.2.8 dirout

```
character(256), dimension(:), allocatable, public mo_common_variables::dirout
```

Referenced by mo_mrm_write_fluxes_states::close(), mo_write_fluxes_states::close(), mo_common_read_config::common_read_config(), mo_mrm_init::config_output(), mo_mrm_write_fluxes_states::createoutputfile(), mo_write_fluxes_states::createoutputfile(), mo_write_ascii::write_configfile(), mo_mrm_write::write_configfile(), and mo_mrm_write::write_daily_obs_sim_discharge().

16.15.2.9 dirrestartout

```
character(256), dimension(:), allocatable, public mo_common_variables::dirrestartout
```

Referenced by mo_common_read_config::common_read_config(), mpr_driver(), mo_mrm_write::mrm_write(), mo_write_ascii::write_configfile(), and mo_mrm_write::write_configfile().

16.15.2.10 filelatlon

```
character(256), dimension(:), allocatable, public mo_common_variables::filelatlon
```

Referenced by mo_common_read_config::common_read_config(), and mo_read_latlon::read_latlon().

16.15.2.11 global_parameters

```
real(dp), dimension(:, :), allocatable, target, public mo_common_variables::global_parameters
```

Referenced by mo_common_mhm_mrm_read_config::check_optimization_settings(), mo_mrm_objective_↩ function_runoff::loglikelihood_evin2013_2(), mo_multi_param_reg::mpr(), mo_mpr_read_config::mpr_read_↩ config(), mrm_driver(), mo_mrm_init::mrm_init(), mo_optimization::optimization(), mo_read_wrapper::read_↩ data(), mo_mrm_read_config::read_mrm_routing_params(), mo_write_ascii::write_configfile(), and mo_mrm_↩ write::write_configfile().

16.15.2.12 global_parameters_name

```
character(256), dimension(:), allocatable, public mo_common_variables::global_parameters_name
```

Referenced by mo_mpr_read_config::mpr_read_config(), mrm_driver(), mo_mrm_read_config::read_mrm_↩ routing_params(), mo_write_ascii::write_configfile(), and mo_mrm_write::write_configfile().

16.15.2.13 history

```
character(1024), public mo_common_variables::history
```

Referenced by mo_common_read_config::common_read_config(), and mo_mrm_river_head::create_output().

16.15.2.14 iflag_cordinate_sys

```
integer(i4), public mo_common_variables::iflag_cordinate_sys
```

Referenced by mo_mrm_river_head::calc_slope(), mo_common_read_config::common_read_config(), mo_grid↩ ::l0_grid_setup(), mo_mrm_net_startup::l11_stream_features(), mo_mrm_mpr::mrm_init_param(), mo_mrm_mpr↩ ::mrm_update_param(), and mo_write_ascii::write_configfile().

16.15.2.15 l0_basin

```
integer(i4), dimension(:), allocatable, public mo_common_variables::l0_basin
```

Referenced by mo_mrm_net_startup::celllength(), mo_common_read_config::common_read_config(), mo_mrm↩ _net_startup::l11_calc_celerity(), mo_mrm_net_startup::l11_flow_direction(), mo_mrm_net_startup::l11_fraction↩ _sealed_floodplain(), mo_mrm_net_startup::l11_link_location(), mo_mrm_net_startup::l11_set_drain_outlet_↩ gauges(), mo_mrm_net_startup::l11_stream_features(), mo_startup::mhm_initialize(), mo_mpr_eval::mpr_eval(),

mo_mpr_startup::mpr_initialize(), mo_mrm_init::mrm_init(), mo_mrm_read_data::mrm_read_l0_data(), mo_↩
mrm_restart::mrm_read_restart_config(), mo_mrm_restart::mrm_write_restart(), mo_read_wrapper::read_data(), mo_common_read_data::read_dem(), mo_common_read_data::read_lcover(), mo_mrm_init::variables_alloc_↩
routing(), mo_write_ascii::write_configfile(), and mo_mrm_write::write_configfile().

16.15.2.16 l0_elev

```
real(dp), dimension(:), allocatable, public mo_common_variables::l0_elev
```

Referenced by mo_mrm_river_head::calc_channel_elevation(), mo_mpr_startup::l0_check_input(), mo_mrm_net↩
_startup::l11_stream_features(), and mo_common_read_data::read_dem().

16.15.2.17 l0_l11_remap

```
type(gridremapper), dimension(:), allocatable, public mo_common_variables::l0_l11_remap
```

16.15.2.18 l0_l1_remap

```
type(gridremapper), dimension(:), allocatable, public mo_common_variables::l0_l1_remap
```

Referenced by mo_mpr_eval::mpr_eval(), mo_mpr_startup::mpr_initialize(), and mo_mrm_init::mrm_init().

16.15.2.19 l0_lcover

```
integer(i4), dimension(:, :), allocatable, public mo_common_variables::l0_lcover
```

Referenced by mo_mpr_startup::l0_check_input(), mo_mrm_net_startup::l11_fraction_sealed_floodplain(), mo_↩
mpr_eval::mpr_eval(), mo_mrm_read_data::mrm_read_l0_data(), and mo_common_read_data::read_lcover().

16.15.2.20 l11_basin

```
integer(i4), dimension(:), allocatable, public mo_common_variables::l11_basin
```

16.15.2.21 l1_l11_remap

```
type(gridremapper), dimension(:), allocatable, public mo_common_variables::l1_l11_remap
```

Referenced by mo_mrm_restart::mrm_write_restart().

16.15.2.22 lc_year_end

```
integer(i4), dimension(:), allocatable, public mo_common_variables::lc_year_end
```

Referenced by mo_common_read_config::common_read_config(), mo_restart::read_restart_states(), mo_common_read_config::set_land_cover_scenes_id(), mo_write_ascii::write_configfile(), mo_mrm_write::write_configfile(), and mo_mpr_restart::write_eff_params().

16.15.2.23 lc_year_start

```
integer(i4), dimension(:), allocatable, public mo_common_variables::lc_year_start
```

Referenced by mo_common_read_config::common_read_config(), mo_restart::read_restart_states(), mo_common_read_config::set_land_cover_scenes_id(), mo_write_ascii::write_configfile(), mo_mrm_write::write_configfile(), and mo_mpr_restart::write_eff_params().

16.15.2.24 lcfilename

```
character(256), dimension(:), allocatable, public mo_common_variables::lcfilename
```

Referenced by mo_common_mhm_mrm_read_config::common_mhm_mrm_read_config(), mo_common_read_config::common_read_config(), mo_common_read_data::read_lcover(), mo_write_ascii::write_configfile(), and mo_mrm_write::write_configfile().

16.15.2.25 level0

```
type(grid), dimension(:), allocatable, target, public mo_common_variables::level0
```

Referenced by mo_mrm_river_head::avg_and_write_timestep(), mo_mrm_river_head::calc_river_head(), mo_mrm_net_startup::celllength(), mo_mrm_river_head::init_masked_zeros_l0(), mo_mpr_startup::l0_check_input(), mo_mrm_init::l0_check_input_routing(), mo_mpr_startup::l0_variable_init(), mo_mrm_net_startup::l11_flow_direction(), mo_mrm_net_startup::l11_fraction_sealed_floodplain(), mo_mrm_net_startup::l11_link_location(), mo_mrm_net_startup::l11_set_drain_outlet_gauges(), mo_mrm_net_startup::l11_stream_features(), mo_startup::mhm_initialize(), mo_mpr_eval::mpr_eval(), mo_mpr_startup::mpr_initialize(), mo_mrm_init::mrm_init(), mo_mrm_read_data::mrm_read_l0_data(), mo_mrm_restart::mrm_read_restart_config(), mo_mrm_restart::mrm_write_restart(), mo_read_wrapper::read_data(), mo_common_read_data::read_dem(), mo_common_read_data::read_lcover(), mo_mrm_river_head::reset_sum(), mo_mrm_init::variables_alloc_routing(), mo_write_ascii::write_configfile(), and mo_mrm_write::write_configfile().

16.15.2.26 level1

```
type(grid), dimension(:), allocatable, target, public mo_common_variables::level1
```

Referenced by mo_write_fluxes_states::createoutputfile(), mo_mrm_net_startup::l11_l1_mapping(), mo_meteo_forcings::meteo_forcings_wrapper(), mo_meteo_forcings::meteo_weights_wrapper(), mo_mhm_eval::mhm_eval(), mo_startup::mhm_initialize(), mo_mpr_eval::mpr_eval(), mo_mpr_startup::mpr_initialize(), mo_mrm_eval::mrm_eval(), mo_mrm_init::mrm_init(), mo_mrm_restart::mrm_read_restart_config(), mo_mrm_read_data::mrm_read_total_runoff(), mo_mrm_restart::mrm_write_restart(), mo_objective_function::objective_et_kge_catchment_avg(), mo_objective_function::objective_kge_q_et(), mo_objective_function::objective_kge_q_rmse_et(), mo_objective_function::objective_kge_q_sm_corr(), mo_objective_function::objective_neutrons_kge_catchment_avg(), mo_objective_function::objective_sm_corr(), mo_objective_function::objective_sm_kge_catchment_avg(), mo_objective_function::objective_sm_pd(), mo_objective_function::objective_sm_sse_standard_score(), mo_read_optional_data::read_evapotranspiration(), mo_read_optional_data::read_neutrons(), mo_restart::read_restart_states(), mo_read_optional_data::read_soil_moisture(), mo_write_ascii::write_configfile(), mo_mrm_write::write_configfile(), mo_mpr_restart::write_mpr_restart_files(), and mo_restart::write_restart_files().

16.15.2.27 mhm_details

```
character(1024), public mo_common_variables::mhm_details
```

Referenced by mo_common_read_config::common_read_config(), and mo_mrm_river_head::create_output().

16.15.2.28 nbasins

```
integer(i4), public mo_common_variables::nbasins
```

Referenced by mo_mrm_river_head::calc_channel_elevation(), mo_common_mhm_mrm_read_config::common_check_resolution(), mo_common_mhm_mrm_read_config::common_mhm_mrm_read_config(), mo_common_read_config::common_read_config(), mo_mrm_init::config_output(), mo_mrm_net_startup::l11_fraction_sealed_floodplain(), mo_mhm_eval::mhm_eval(), mo_startup::mhm_initialize(), mo_mhm_read_config::mhm_read_config(), mo_mpr_eval::mpr_eval(), mo_mpr_startup::mpr_initialize(), mo_mpr_read_config::mpr_read_config(), mo_mrm_eval::mrm_eval(), mo_mrm_init::mrm_init(), mo_mrm_mpr::mrm_init_param(), mo_mrm_read_config::mrm_read_config(), mo_mrm_read_data::mrm_read_discharge(), mo_mrm_read_data::mrm_read_l0_data(), mo_mrm_restart::mrm_read_restart_config(), mo_mrm_write::mrm_write(), mo_objective_function::objective_et_kge_catchment_avg(), mo_objective_function::objective_kge_q_et(), mo_objective_function::objective_kge_q_rmse_et(), mo_objective_function::objective_kge_q_rmse_tws(), mo_objective_function::objective_kge_q_sm_corr(), mo_objective_function::objective_neutrons_kge_catchment_avg(), mo_objective_function::objective_sm_corr(), mo_objective_function::objective_sm_kge_catchment_avg(), mo_objective_function::objective_sm_pd(), mo_objective_function::objective_sm_sse_standard_score(), mo_meteo_forcings::prepare_meteo_forcings_data(), mo_read_optional_data::read_basin_avg_tws(), mo_read_wrapper::read_data(), mo_common_read_data::read_dem(), mo_common_read_data::read_lcover(), mo_common_read_config::set_land_cover_scenes_id(), mo_write_ascii::write_configfile(), mo_mrm_write::write_configfile(), and mo_mrm_write::write_daily_obs_sim_discharge().

16.15.2.29 nlcoverscene

```
integer(i4), public mo_common_variables::nlcoverscene
```

Referenced by mo_common_read_config::common_read_config(), mo_mpr_startup::init_eff_params(), mo_mpr_startup::l0_check_input(), mo_mrm_net_startup::l11_fraction_sealed_floodplain(), mo_mrm_restart::mrm_read_restart_states(), mo_mrm_restart::mrm_write_restart(), mo_common_read_data::read_lcover(), mo_restart::read_restart_states(), mo_common_read_config::set_land_cover_scenes_id(), mo_write_ascii::write_configfile(), mo_mrm_write::write_configfile(), mo_mpr_restart::write_mpr_restart_files(), and mo_restart::write_restart_files().

16.15.2.30 nprocesses

```
integer(i4), parameter, public mo_common_variables::nprocesses = 10
```

Referenced by mo_common_read_config::common_read_config(), and mo_write_ascii::write_optinamelist().

16.15.2.31 nuniquel0basins

```
integer(i4), public mo_common_variables::nuniquel0basins
```

Referenced by mo_common_read_config::common_read_config(), and mo_common_read_data::read_dem().

16.15.2.32 processmatrix

```
integer(i4), dimension(nprocesses, 3), public mo_common_variables::processmatrix
```

Referenced by mo_common_read_config::common_read_config(), mo_startup::constants_init(), mo_mrm_net_startup::l11_stream_features(), mo_mhm_eval::mhm_eval(), mo_mhm_read_config::mhm_read_config(), mo_multi_param_reg::mpr(), mo_mpr_read_config::mpr_read_config(), mo_mrm_eval::mrm_eval(), mo_mrm_init::mrm_init(), mo_mrm_mpr::mrm_init_param(), mo_mrm_read_config::mrm_read_config(), mo_mrm_read_data::mrm_read_l0_data(), mo_mrm_restart::mrm_read_restart_config(), mo_mrm_mpr::mrm_update_param(), mo_mrm_write::mrm_write_optinamelist(), mo_mrm_restart::mrm_write_restart(), mo_meteo_forcings::prepare_meteo_forcings_data(), mo_read_wrapper::read_data(), mo_mrm_read_config::read_mrm_routing_params(), mo_restart::read_restart_states(), mo_write_ascii::write_configfile(), mo_mrm_write::write_configfile(), and mo_mpr_restart::write_eff_params().

16.15.2.33 project_details

```
character(1024), public mo_common_variables::project_details
```

Referenced by mo_common_read_config::common_read_config(), and mo_mrm_river_head::create_output().

16.15.2.34 resolutionhydrology

```
real(dp), dimension(:), allocatable, public mo_common_variables::resolutionhydrology
```

Referenced by mo_common_mhm_mrm_read_config::common_check_resolution(), mo_common_read_config::common_read_config(), mo_mhm_eval::mhm_eval(), mo_mpr_startup::mpr_initialize(), mo_mrm_eval::mrm_eval(), mo_mrm_init::mrm_init(), mo_common_read_data::read_dem(), mo_write_ascii::write_configfile(), and mo_mrm_write::write_configfile().

16.15.2.35 setup_description

```
character(1024), public mo_common_variables::setup_description
```

Referenced by mo_common_read_config::common_read_config(), and mo_mrm_river_head::create_output().

16.15.2.36 simulation_type

```
character(1024), public mo_common_variables::simulation_type
```

Referenced by mo_common_read_config::common_read_config(), and mo_mrm_river_head::create_output().

16.15.2.37 write_restart

```
logical, public mo_common_variables::write_restart
```

Referenced by mo_common_read_config::common_read_config(), mhm_driver(), mpr_driver(), mo_mrm_write::mrm_write(), mo_write_ascii::write_configfile(), and mo_mrm_write::write_configfile().

16.16 mo_constants Module Reference

Provides computational, mathematical, physical, and file constants.

Variables

- real(dp), parameter pi_dp = 3.14159265358979323846264338327950288419 7_dp
  Pi in double precision.
- real(sp), parameter pi_sp = 3.14159265358979323846264338327950288419 7_sp
  Pi in single precision.
- real(dp), parameter pio2_dp = 1.57079632679489661923132169163975144209858_dp
  Pi/2 in double precision.
- real(sp), parameter pio2_sp = 1.57079632679489661923132169163975144209858_sp
  Pi/2 in single precision.
- real(dp), parameter twopi_dp = 6.28318530717958647692528676655900576839 4_dp
  2∗Pi in double precision
- real(sp), parameter twopi_sp = 6.28318530717958647692528676655900576839 4_sp
  2∗Pi in single precision
- real(dp), parameter sqrt2_dp = 1.41421356237309504880168872420969807856967_dp
  Square root of 2 in double precision.
- real(sp), parameter sqrt2_sp = 1.41421356237309504880168872420969807856967_sp
  Square root of 2 in single precision.
- real(dp), parameter twothird_dp = 0.66666666666666666666666666666666666667_dp
  2/3 in double precision
- real(sp), parameter twothird_sp = 0.66666666666666666666666666666666666667_sp
  2/3 in single precision
- real(dp), parameter deg2rad_dp = PI_dp / 180._dp
  degree to radian conversion (pi/180) in double precision
- real(sp), parameter deg2rad_sp = PI_sp / 180._sp
  degree to radian conversion (pi/180) in double precision
- real(dp), parameter rad2deg_dp = 180._dp / PI_dp
  radian to conversion (180/pi) in double precision
- real(sp), parameter rad2deg_sp = 180._sp / PI_sp
  radian to degree conversion (180/pi) in single precision
- real(sp), parameter, public secday_sp = 86400.0_sp
  Time conversion Seconds per day [s] in single precision.
- real(dp), parameter, public secday_dp = 86400.0_dp
- real(dp), parameter, public dayhours = 24.0_dp
- real(dp), parameter, public yearmonths = 12.0_dp
- real(dp), parameter, public yeardays = 365.0_dp
- real(dp), parameter, public daysecs = 86400.0_dp
- real(dp), parameter psychro_dp = 0.0646_dp
  Psychrometric constant [kPa K^-1] in double precision.
- real(sp), parameter psychro_sp = 0.0646_sp
  Psychrometric constant [kPa K^-1] in sibgle precision.
- real(dp), parameter gravity_dp = 9.81_dp
  Gravity accelaration [m^2 s^-1] in double precision.
- real(sp), parameter gravity_sp = 9.81_sp
  Gravity accelaration [m^2 s^-1] in single precision.
- real(dp), parameter solarconst_dp = 1367._dp
  Solar constant in [J m^-2 s^-1] in double precision.

- real(sp), parameter solarconst_sp = 1367._sp

  Solar constant in [J m^-2 s^-1] in single precision.
- real(dp), parameter specheatet_dp = 2.45e06_dp

  Specific heat for vaporization of water in [J m-2 mm-1] in double precision.
- real(sp), parameter specheatet_sp = 2.45e06_sp

  Specific heat for vaporization of water in [J m-2 mm-1] in single precision.
- real(dp), parameter t0_dp = 273.15_dp

  Standard temperature [K] in double precision.
- real(sp), parameter t0_sp = 273.15_sp

  Standard temperature [K] in single precision.
- real(dp), parameter sigma_dp = 5.67e-08_dp

  Stefan-Boltzmann constant [W m^-2 K^-4] in double precision.
- real(sp), parameter sigma_sp = 5.67e-08_sp

  Stefan-Boltzmann constant [W m^-2 K^-4] in single precision.
- real(sp), parameter radiusearth_sp = 6371228._sp
- real(dp), parameter radiusearth_dp = 6371228._dp
- real(dp), parameter p0_dp = 101325._dp

  standard atmospehere Standard pressure [Pa] in double precision
- real(sp), parameter p0_sp = 101325._sp

  Standard pressure [Pa] in single precision.
- real(dp), parameter rho0_dp = 1.225_dp

  standard density [kg m^-3] in double precision
- real(sp), parameter rho0_sp = 1.225_sp

  standard density [kg m^-3] in single precision
- real(dp), parameter cp0_dp = 1005.0_dp

  specific heat capacity of air [J kg^-1 K^-1] in double precision
- real(sp), parameter cp0_sp = 1005.0_sp

  specific heat capacity of air [J kg^-1 K^-1] in single precision
- real(dp), parameter pi_d = 3.14159265358979323846264338327950288419 7_dp

  Pi in double precision.
- real(sp), parameter pi = 3.14159265358979323846264338327950288419 7_sp

  Pi in single precision.
- real(dp), parameter pio2_d = 1.57079632679489661923132169163975144209858_dp

  Pi/2 in double precision.
- real(sp), parameter pio2 = 1.57079632679489661923132169163975144209858_sp

  Pi/2 in single precision.
- real(dp), parameter twopi_d = 6.28318530717958647692528676655900576839 4_dp

  2∗Pi in double precision
- real(sp), parameter twopi = 6.28318530717958647692528676655900576839 4_sp

  2∗Pi in single precision
- real(dp), parameter sqrt2_d = 1.41421356237309504880168872420969807856967_dp

  Square root of 2 in double precision.
- real(sp), parameter sqrt2 = 1.41421356237309504880168872420969807856967_sp

  Square root of 2 in single precision.
- real(dp), parameter euler_d = 0.57721566490153286060651209008240243104 22_dp

  Euler"s constant in double precision.
- real(sp), parameter euler = 0.57721566490153286060651209008240243104 22_sp

  Euler"s constant in single precision.
- integer, parameter nin = input_unit

  Standard input file unit.
- integer, parameter nout = output_unit

Standard output file unit.

- integer, parameter nerr = error_unit

  Standard error file unit.

- integer, parameter nnml = 100

  Standard file unit for namelist.

16.16.1 Detailed Description

Provides computational, mathematical, physical, and file constants.

Provides computational constants like epsilon, mathematical constants such as Pi, physical constants such as the Stefan-Boltzmann constant, and file units for some standard streams such as standard in.

Author

Matthias Cuntz

Date

Nov 2011

16.16.2 Variable Documentation

16.16.2.1 cp0_dp

```
real(dp), parameter mo_constants::cp0_dp = 1005.0_dp
```

specific heat capacity of air [J kg^-1 K^-1] in double precision

Referenced by mo_pet::pet_penman().

16.16.2.2 cp0_sp

```
real(sp), parameter mo_constants::cp0_sp = 1005.0_sp
```

specific heat capacity of air [J kg^-1 K^-1] in single precision

16.16.2.3 dayhours

```
real(dp), parameter, public mo_constants::dayhours = 24.0_dp
```

Referenced by mo_read_forcing_nc::get_time_vector_and_select().

16.16.2.4 daysecs

```
real(dp), parameter, public mo_constants::daysecs = 86400.0_dp
```

Referenced by mo_read_forcing_nc::get_time_vector_and_select().

16.16.2.5 deg2rad_dp

```
real(dp), parameter mo_constants::deg2rad_dp = PI_dp / 180._dp
```

degree to radian conversion (pi/180) in double precision

Referenced by mo_pet::pet_hargreaves().

16.16.2.6 deg2rad_sp

```
real(sp), parameter mo_constants::deg2rad_sp = PI_sp / 180._sp
```

degree to radian conversion (pi/180) in double precision

16.16.2.7 euler

```
real(sp), parameter mo_constants::euler = 0.57721566490153286060651209008240243104 22_sp
```

Euler"s constant in single precision.

16.16.2.8 euler_d

```
real(dp), parameter mo_constants::euler_d = 0.57721566490153286060651209008240243104 22_dp
```

Euler"s constant in double precision.

16.16.2.9 gravity_dp

```
real(dp), parameter mo_constants::gravity_dp = 9.81_dp
```

Gravity accelaration [m^2 s^-1] in double precision.

16.16.2.10 gravity_sp

```
real(sp), parameter mo_constants::gravity_sp = 9.81_sp
```

Gravity accelaration [m^2 s^-1] in single precision.

16.16.2.11 nerr

```
integer, parameter mo_constants::nerr = error_unit
```

Standard error file unit.

16.16.2.12 nin

```
integer, parameter mo_constants::nin = input_unit
```

Standard input file unit.

16.16.2.13 nnml

```
integer, parameter mo_constants::nnml = 100
```

Standard file unit for namelist.

16.16.2.14 nout

```
integer, parameter mo_constants::nout = output_unit
```

Standard output file unit.

Referenced by mo_message::message().

16.16.2.15 p0_dp

```
real(dp), parameter mo_constants::p0_dp = 101325._dp
```

standard atmospehere Standard pressure [Pa] in double precision

16.16.2.16 p0_sp

```
real(sp), parameter mo_constants::p0_sp = 101325._sp
```

Standard pressure [Pa] in single precision.

16.16.2.17 pi

```
real(sp), parameter mo_constants::pi = 3.14159265358979323846264338327950288419 7_sp
```

Pi in single precision.

16.16.2.18 pi_d

```
real(dp), parameter mo_constants::pi_d = 3.14159265358979323846264338327950288419 7_dp
```

Pi in double precision.

Referenced by mo_pet::extraterr_rad_approx().

16.16.2.19 pi_dp

```
real(dp), parameter mo_constants::pi_dp = 3.14159265358979323846264338327950288419 7_dp
```

Pi in double precision.

Referenced by mo_neutrons::cosmic(), mo_corr::fourrow_dp(), mo_corr::fourrow_sp(), mo_mrm_objective_↩ function_runoff::loglikelihood_evin2013_2(), mo_neutrons::lookupintegral(), mo_neutrons::oldintegration(), mo_↩ mrm_objective_function_runoff::parameter_regularization(), and mo_neutrons::tabularintegralafast().

16.16.2.20 pi_sp

```
real(sp), parameter mo_constants::pi_sp = 3.14159265358979323846264338279502884197_sp
```

Pi in single precision.

16.16.2.21 pio2

```
real(sp), parameter mo_constants::pio2 = 1.57079632679489661923132169163975144209858_sp
```

Pi/2 in single precision.

16.16.2.22 pio2_d

```
real(dp), parameter mo_constants::pio2_d = 1.57079632679489661923132169163975144209858_dp
```

Pi/2 in double precision.

16.16.2.23 pio2_dp

```
real(dp), parameter mo_constants::pio2_dp = 1.57079632679489661923132169163975144209858_dp
```

Pi/2 in double precision.

16.16.2.24 pio2_sp

```
real(sp), parameter mo_constants::pio2_sp = 1.57079632679489661923132169163975144209858_sp
```

Pi/2 in single precision.

16.16.2.25 psychro_dp

```
real(dp), parameter mo_constants::psychro_dp = 0.0646_dp
```

Psychrometric constant [kPa K^-1] in double precision.

Referenced by mo_pet::pet_penman(), and mo_pet::pet_priestly().

16.16.2.26 psychro_sp

```
real(sp), parameter mo_constants::psychro_sp = 0.0646_sp
```

Psychrometric constant [kPa K^-1] in sibgle precision.

16.16.2.27 rad2deg_dp

```
real(dp), parameter mo_constants::rad2deg_dp = 180._dp / PI_dp
```

radian to conversion (180/pi) in double precision

16.16.2.28 rad2deg_sp

```
real(sp), parameter mo_constants::rad2deg_sp = 180._sp / PI_sp
```

radian to degree conversion (180/pi) in single precision

16.16.2.29 radiusearth_dp

```
real(dp), parameter mo_constants::radiusearth_dp = 6371228._dp
```

Referenced by mo_mrm_net_startup::get_distance_two_lat_lon_points(), and mo_grid::l0_grid_setup().

16.16.2.30 radiusearth_sp

```
real(sp), parameter mo_constants::radiusearth_sp = 6371228._sp
```

16.16.2.31 rho0_dp

```
real(dp), parameter mo_constants::rho0_dp = 1.225_dp
```

standard density [kg m^-3] in double precision

Referenced by mo_pet::pet_penman().

16.16.2.32 rho0_sp

```
real(sp), parameter mo_constants::rho0_sp = 1.225_sp
```

standard density [kg m^-3] in single precision

16.16.2.33 secday_dp

```
real(dp), parameter, public mo_constants::secday_dp = 86400.0_dp
```

16.16.2.34 secday_sp

```
real(sp), parameter, public mo_constants::secday_sp = 86400.0_sp
```

Time conversion Seconds per day [s] in single precision.

16.16.2.35 sigma_dp

```
real(dp), parameter mo_constants::sigma_dp = 5.67e-08_dp
```

Stefan-Boltzmann constant [W m^-2 K^-4] in double precision.

16.16.2.36 sigma_sp

```
real(sp), parameter mo_constants::sigma_sp = 5.67e-08_sp
```

Stefan-Boltzmann constant [W m^-2 K^-4] in single precision.

16.16.2.37 solarconst_dp

```
real(dp), parameter mo_constants::solarconst_dp = 1367._dp
```

Solar constant in [J m^-2 s^-1] in double precision.

Referenced by mo_pet::extraterr_rad_approx().

16.16.2.38 solarconst_sp

```
real(sp), parameter mo_constants::solarconst_sp = 1367._sp
```

Solar constant in [J m^-2 s^-1] in single precision.

16.16.2.39 specheatet_dp

```
real(dp), parameter mo_constants::specheatet_dp = 2.45e06_dp
```

Specific heat for vaporization of water in [J m-2 mm-1] in double precision.

Referenced by mo_pet::extraterr_rad_approx(), mo_pet::pet_penman(), and mo_pet::pet_priestly().

16.16.2.40 specheatet_sp

```
real(sp), parameter mo_constants::specheatet_sp = 2.45e06_sp
```

Specific heat for vaporization of water in [J m-2 mm-1] in single precision.

16.16.2.41 sqrt2

```
real(sp), parameter mo_constants::sqrt2 = 1.41421356237309504880168872420969807856967_sp
```

Square root of 2 in single precision.

16.16.2.42 sqrt2_d

```
real(dp), parameter mo_constants::sqrt2_d = 1.41421356237309504880168872420969807856967_dp
```

Square root of 2 in double precision.

16.16.2.43 sqrt2_dp

```
real(dp), parameter mo_constants::sqrt2_dp = 1.41421356237309504880168872420969807856967_dp
```

Square root of 2 in double precision.

Referenced by mo_mrm_net_startup::celllength().

16.16.2.44 sqrt2_sp

```
real(sp), parameter mo_constants::sqrt2_sp = 1.41421356237309504880168872420969807856967_sp
```

Square root of 2 in single precision.

16.16.2.45 t0_dp

```
real(dp), parameter mo_constants::t0_dp = 273.15_dp
```

Standard temperature [K] in double precision.

16.16.2.46 t0_sp

```
real(sp), parameter mo_constants::t0_sp = 273.15_sp
```

Standard temperature [K] in single precision.

16.16.2.47 twopi

```
real(sp), parameter mo_constants::twopi = 6.28318530717958647692528676655900576839 4_sp
```

2∗Pi in single precision

16.16.2.48 twopi_d

```
real(dp), parameter mo_constants::twopi_d = 6.28318530717958647692528676655900576839 4_dp
```

2∗Pi in double precision

Referenced by mo_pet::extraterr_rad_approx().

16.16.2.49 twopi_dp

```
real(dp), parameter mo_constants::twopi_dp = 6.28318530717958647692528676655900576839 4_dp
```

2∗Pi in double precision

Referenced by mo_corr::four1_dp(), mo_corr::four1_sp(), mo_mrm_net_startup::get_distance_two_lat_lon_↩ points(), mo_grid::l0_grid_setup(), and mo_corr::zroots_unity_dp().

16.16.2.50 twopi_sp

```
real(sp), parameter mo_constants::twopi_sp = 6.28318530717958647692528676655900576839 4_sp
```

2∗Pi in single precision

Referenced by mo_corr::zroots_unity_sp().

16.16.2.51 twothird_dp

```
real(dp), parameter mo_constants::twothird_dp = 0.66666666666666666666666666666666666667_dp
```

2/3 in double precision

Referenced by mo_canopy_interc::canopy_interc().

16.16.2.52 twothird_sp

```
real(sp), parameter mo_constants::twothird_sp = 0.66666666666666666666666666666666666667_sp
```

2/3 in single precision

16.16.2.53 yeardays

```
real(dp), parameter, public mo_constants::yeardays = 365.0_dp
```

Referenced by mo_read_forcing_nc::get_time_vector_and_select().

16.16.2.54 yearmonths

```
real(dp), parameter, public mo_constants::yearmonths = 12.0_dp
```

16.17 mo_corr Module Reference

Data Types

- interface arth
- interface autocoeffk
- interface autocorr
- interface corr

- interface crosscoeffk
- interface crosscorr
- interface four1
- interface fourrow
- interface realft
- interface swap

Functions/Subroutines

- real(sp) function, dimension(n) arth_sp (first, increment, n)
- real(dp) function, dimension(n) arth_dp (first, increment, n)
- integer(i4) function, dimension(n) arth_i4 (first, increment, n)
- real(dp) function autocoeffk_dp (x, k, mask)
- real(sp) function autocoeffk_sp (x, k, mask)
- real(dp) function, dimension(size(k)) autocoeffk_1d_dp (x, k, mask)
- real(sp) function, dimension(size(k)) autocoeffk_1d_sp (x, k, mask)
- real(dp) function autocorr_dp (x, k, mask)
- real(sp) function autocorr_sp (x, k, mask)
- real(dp) function, dimension(size(k)) autocorr_1d_dp (x, k, mask)
- real(sp) function, dimension(size(k)) autocorr_1d_sp (x, k, mask)
- real(dp) function, dimension(size(data1)) corr_dp (data1, data2, nadjust, nhigh, nwin)
- real(sp) function, dimension(size(data1)) corr_sp (data1, data2, nadjust, nhigh, nwin)
- real(dp) function crosscoeffk_dp (x, y, k, mask)
- real(sp) function crosscoeffk_sp (x, y, k, mask)
- real(dp) function crosscorr_dp (x, y, k, mask)
- real(sp) function crosscorr_sp (x, y, k, mask)
- subroutine four1_sp (data, isign)
- subroutine four1_dp (data, isign)
- subroutine fourrow_sp (data, isign)
- subroutine fourrow_dp (data, isign)
- subroutine realft_dp (data, isign, zdata)
- subroutine realft_sp (data, isign, zdata)
- subroutine swap_1d_spc (a, b)
- subroutine swap_1d_dpc (a, b)
- complex(dpc) function, dimension(nn) zroots_unity_dp (n, nn)
- complex(spc) function, dimension(nn) zroots_unity_sp (n, nn)

Variables

- integer(i4), parameter npar_arth = 16
- integer(i4), parameter npar2_arth = 8

16.17.1 Function/Subroutine Documentation

16.17.1.1 arth_dp()

```
real(dp) function, dimension(n) mo_corr::arth_dp (
            real(dp), intent(in) first,
            real(dp), intent(in) increment,
            integer(i4), intent(in) n )  [private]
```

References npar2_arth, and npar_arth.

16.17.1.2 arth_i4()

```
integer(i4) function, dimension(n) mo_corr::arth_i4 (
            integer(i4), intent(in) first,
            integer(i4), intent(in) increment,
            integer(i4), intent(in) n )  [private]
```

References npar2_arth, and npar_arth.

16.17.1.3 arth_sp()

```
real(sp) function, dimension(n) mo_corr::arth_sp (
            real(sp), intent(in) first,
            real(sp), intent(in) increment,
            integer(i4), intent(in) n )  [private]
```

References npar2_arth, and npar_arth.

16.17.1.4 autocoeffk_1d_dp()

```
real(dp) function, dimension(size(k)) mo_corr::autocoeffk_1d_dp (
            real(dp), dimension(:), intent(in) x,
            integer(i4), dimension(:), intent(in) k,
            logical, dimension(:), intent(in), optional mask )  [private]
```

16.17.1.5 autocoeffk_1d_sp()

```
real(sp) function, dimension(size(k)) mo_corr::autocoeffk_1d_sp (
            real(sp), dimension(:), intent(in) x,
            integer(i4), dimension(:), intent(in) k,
            logical, dimension(:), intent(in), optional mask )  [private]
```

16.17.1.6 autocoeffk_dp()

```
real(dp) function mo_corr::autocoeffk_dp (
            real(dp), dimension(:), intent(in) x,
            integer(i4), intent(in) k,
            logical, dimension(:), intent(in), optional mask )  [private]
```

16.17.1.7 autocoeffk_sp()

```
real(sp) function mo_corr::autocoeffk_sp (
            real(sp), dimension(:), intent(in) x,
            integer(i4), intent(in) k,
            logical, dimension(:), intent(in), optional mask )  [private]
```

16.17.1.8 autocorr_1d_dp()

```
real(dp) function, dimension(size(k)) mo_corr::autocorr_1d_dp (
            real(dp), dimension(:), intent(in) x,
            integer(i4), dimension(:), intent(in) k,
            logical, dimension(:), intent(in), optional mask )  [private]
```

16.17.1.9 autocorr_1d_sp()

```
real(sp) function, dimension(size(k)) mo_corr::autocorr_1d_sp (
            real(sp), dimension(:), intent(in) x,
            integer(i4), dimension(:), intent(in) k,
            logical, dimension(:), intent(in), optional mask )  [private]
```

16.17.1.10 autocorr_dp()

```
real(dp) function mo_corr::autocorr_dp (
            real(dp), dimension(:), intent(in) x,
            integer(i4), intent(in) k,
            logical, dimension(:), intent(in), optional mask )  [private]
```

16.17.1.11 autocorr_sp()

```
real(sp) function mo_corr::autocorr_sp (
            real(sp), dimension(:), intent(in) x,
            integer(i4), intent(in) k,
            logical, dimension(:), intent(in), optional mask )  [private]
```

16.17.1.12 corr_dp()

```
real(dp) function, dimension(size(data1)) mo_corr::corr_dp (
            real(dp), dimension(:), intent(in) data1,
            real(dp), dimension(:), intent(in) data2,
            integer(i4), intent(out), optional nadjust,
            integer(i4), intent(in), optional nhigh,
            integer(i4), intent(in), optional nwin )  [private]
```

16.17.1.13 corr_sp()

```
real(sp) function, dimension(size(data1)) mo_corr::corr_sp (
            real(sp), dimension(:), intent(in) data1,
            real(sp), dimension(:), intent(in) data2,
            integer(i4), intent(out), optional nadjust,
            integer(i4), intent(in), optional nhigh,
            integer(i4), intent(in), optional nwin )  [private]
```

16.17.1.14 crosscoeffk_dp()

```
real(dp) function mo_corr::crosscoeffk_dp (
            real(dp), dimension(:), intent(in) x,
            real(dp), dimension(:), intent(in) y,
            integer(i4), intent(in) k,
            logical, dimension(:), intent(in), optional mask )  [private]
```

16.17.1.15 crosscoeffk_sp()

```
real(sp) function mo_corr::crosscoeffk_sp (
            real(sp), dimension(:), intent(in) x,
            real(sp), dimension(:), intent(in) y,
            integer(i4), intent(in) k,
            logical, dimension(:), intent(in), optional mask )  [private]
```

16.17.1.16 crosscorr_dp()

```
real(dp) function mo_corr::crosscorr_dp (
            real(dp), dimension(:), intent(in) x,
            real(dp), dimension(:), intent(in) y,
            integer(i4), intent(in) k,
            logical, dimension(:), intent(in), optional mask )  [private]
```

16.17.1.17 crosscorr_sp()

```
real(sp) function mo_corr::crosscorr_sp (
            real(sp), dimension(:), intent(in) x,
            real(sp), dimension(:), intent(in) y,
            integer(i4), intent(in) k,
            logical, dimension(:), intent(in), optional mask )  [private]
```

16.17.1.18 four1_dp()

```
subroutine mo_corr::four1_dp (
            complex(dpc), dimension(:), intent(inout) data,
            integer(i4), intent(in) isign )  [private]
```

References mo_kind::dpc, and mo_constants::twopi_dp.

16.17.1.19 four1_sp()

```
subroutine mo_corr::four1_sp (
            complex(spc), dimension(:), intent(inout) data,
            integer(i4), intent(in) isign )  [private]
```

References mo_kind::dpc, mo_kind::spc, and mo_constants::twopi_dp.

16.17.1.20 fourrow_dp()

```
subroutine mo_corr::fourrow_dp (
            complex(dpc), dimension(:, :), intent(inout) data,
            integer(i4), intent(in) isign )  [private]
```

References mo_kind::dpc, and mo_constants::pi_dp.

16.17.1.21 fourrow_sp()

```
subroutine mo_corr::fourrow_sp (
            complex(spc), dimension(:, :), intent(inout) data,
            integer(i4), intent(in) isign )  [private]
```

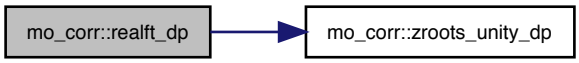
References mo_kind::dpc, mo_constants::pi_dp, and mo_kind::spc.

16.17.1.22 realft_dp()

```
subroutine mo_corr::realft_dp (
            real(dp), dimension(:), intent(inout) data,
            integer(i4), intent(in) isign,
            complex(dpc), dimension(:), optional, target zdata )  [private]
```

References mo_kind::dpc, and zroots_unity_dp().

Here is the call graph for this function:

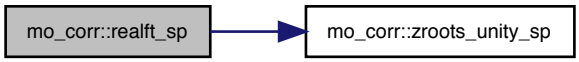


16.17.1.23 realft_sp()

```
subroutine mo_corr::realft_sp (
            real(sp), dimension(:), intent(inout) data,
            integer(i4), intent(in) isign,
            complex(spc), dimension(:), optional, target zdata )  [private]
```

References mo_kind::spc, and zroots_unity_sp().

Here is the call graph for this function:



16.17.1.24 swap_1d_dpc()

```
subroutine mo_corr::swap_1d_dpc (
             complex(dpc), dimension(:), intent(inout) a,
             complex(dpc), dimension(:), intent(inout) b )  [private]
```

16.17.1.25 swap_1d_spc()

```
subroutine mo_corr::swap_1d_spc (
             complex(spc), dimension(:), intent(inout) a,
             complex(spc), dimension(:), intent(inout) b )  [private]
```

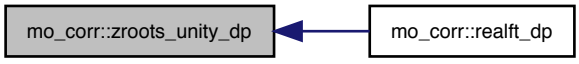
16.17.1.26 zroots_unity_dp()

```
complex(dpc) function, dimension(nn) mo_corr::zroots_unity_dp (
             integer(i4), intent(in) n,
             integer(i4), intent(in) nn )  [private]
```

References mo_kind::dpc, and mo_constants::twopi_dp.

Referenced by realft_dp().

Here is the caller graph for this function:



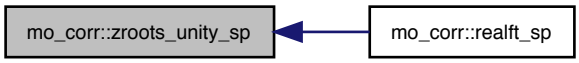
16.17.1.27 zroots_unity_sp()

```
complex(spc) function, dimension(nn) mo_corr::zroots_unity_sp (
             integer(i4), intent(in) n,
             integer(i4), intent(in) nn )  [private]
```

References mo_kind::spc, and mo_constants::twopi_sp.

Referenced by realft_sp().

Here is the caller graph for this function:



16.17.2 Variable Documentation

16.17.2.1 npar2_arth

```
integer(i4), parameter mo_corr::npar2_arth = 8  [private]
```

Referenced by arth_dp(), arth_i4(), and arth_sp().

16.17.2.2 npar_arth

```
integer(i4), parameter mo_corr::npar_arth = 16  [private]
```

Referenced by arth_dp(), arth_i4(), and arth_sp().

16.18 mo_dds Module Reference

Dynamically Dimensioned Search (DDS)

Functions/Subroutines

- real(dp) function, dimension(size(pini)), public dds (eval, obj_func, pini, prange, r, seed, maxiter, maxit, mask, tmp_file, funcbest, history)

  DDS.
- real(dp) function, dimension(size(pini)), public mdds (eval, obj_func, pini, prange, seed, maxiter, maxit, mask, tmp_file, funcbest, history)

  MDDS.
- subroutine neigh_value (x_cur, x_min, x_max, r, new_value)

16.18.1 Detailed Description

Dynamically Dimensioned Search (DDS)

This module provides routines for Dynamically Dimensioned Search (DDS) of Tolson and Shoemaker (2007). It searches the minimum or maximum of a user-specified function, using an n-dimensional continuous global optimization algorithm (DDS).

Authors

Original by Bryan Tolson and later modified by Rohini Kumar. Matthias Cuntz and Juliane Mai for the module, MDDS, etc.

Date

Jul 2012

16.18.2 Function/Subroutine Documentation

16.18.2.1 dds()

```
real(dp) function, dimension(size(pini)), public mo_dds::dds (
             procedure(eval_interface), intent(in), pointer eval,
             procedure(objective_interface), intent(in), pointer obj_func,
             real(dp), dimension(:), intent(in) pini,
             real(dp), dimension(:, :), intent(in) prange,
             real(dp), intent(in), optional r,
             integer(i8), intent(in), optional seed,
             integer(i8), intent(in), optional maxiter,
             logical, intent(in), optional maxit,
             logical, dimension(:), intent(in), optional mask,
             character(len = *), intent(in), optional tmp_file,
             real(dp), intent(out), optional funcbest,
             real(dp), dimension(:), intent(out), optional, allocatable history )
```

DDS.

Searches Minimum or Maximum of a user-specified function using Dynamically Dimensioned Search (DDS).

DDS is an n-dimensional continuous global optimization algorithm. It is coded as a minimizer but one can give maxit=True in a maximization problem, so that the algorithm minimizes the negative of the objective function F=(-1∗F). The function to be minimized is the first argument of DDS and must be defined as

```
function func(p)
  use mo_kind, only: dp
  implicit none
  real(dp), dimension(:), intent(in) :: p
  real(dp) :: func
end function func
```

Parameters

in	*real(dp) :: obj_func(p)*	Function on which to search the minimum
in	*real(dp) :: pini(:)*	inital value of decision variables
in	*real(dp) :: prange(size(pini),2)*	Min/max range of decision variables
in	*real(dp), optional :: r*	DDS perturbation parameter (default: 0.2)

Parameters

in	*integer(i8), optional :: seed*	User seed to initialise the random number generator (default: None)
in	*integer(i8), optional :: maxiter*	Maximum number of iteration or function evaluation (default: 1000)
in	*logical, optional :: maxit*	Maximization (.True.) or minimization (.False.) of function (default: .False.)
in	*logical, optional :: mask(size(pini))*	parameter to be optimized (true or false) (default: .True.)
in	*character(len=∗) , optional :: tmp_file*	file with temporal output
out	*real(dp), optional :: funcbest*	the best value of the function.
out	*real(dp), optional, allocatable :: history(:)*	the history of best function values, history(maxiter)=funcbest allocatable only to be in correspondance with other optimization routines

Returns

real(dp) :: DDS — The parameters of the point which is estimated to minimize the function.

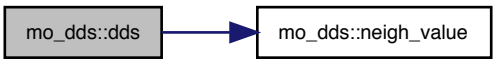
Author

Written original Bryan Tolson - DDS v1.1
Modified Rohini Kumar, Matthias Cuntz, Juliane Mai

References mo_kind::dp, mo_kind::i4, mo_kind::i8, and neigh_value().

Referenced by mo_optimization::optimization().

Here is the call graph for this function:



Here is the caller graph for this function:

16.18.2.2 mdds()

```
real(dp) function, dimension(size(pini)), public mo_dds::mdds (
             procedure(eval_interface), intent(in), pointer eval,
             procedure(objective_interface), intent(in), pointer obj_func,
             real(dp), dimension(:), intent(in) pini,
             real(dp), dimension(:, :), intent(in) prange,
             integer(i8), intent(in), optional seed,
             integer(i8), intent(in), optional maxiter,
             logical, intent(in), optional maxit,
             logical, dimension(:), intent(in), optional mask,
             character(len = *), intent(in), optional tmp_file,
             real(dp), intent(out), optional funcbest,
             real(dp), dimension(:), intent(out), optional, allocatable history )
```

MDDS.

Searches Minimum or Maximum of a user-specified function using the Modified Dynamically Dimensioned Search (DDS). DDS is an n-dimensional continuous global optimization algorithm. It is coded as a minimizer but one can give maxit=True in a maximization problem, so that the algorithm minimizes the negative of the objective function F=(-1∗F).

The function to be minimized is the first argument of DDS and must be defined as

```
function func(p)
  use mo_kind, only: dp
  implicit none
  real(dp), dimension(:), intent(in) :: p
  real(dp) :: func
end function func
```

MDDS extents normal DDS by a continuous reduction of the DDS pertubation parameter r from 0.3 to 0.05, and by allowing a larger function value with a certain probablity.

Parameters

in	*real(dp) :: obj_func(p)*	Function on which to search the minimum
in	*real(dp) :: pini(:)*	inital value of decision variables
in	*real(dp) :: prange(size(pini),2)*	Min/max range of decision variables
in	*integer(i8), optional :: seed*	User seed to initialise the random number generator (default: None)
in	*integer(i8), optional :: maxiter*	Maximum number of iteration or function evaluation (default: 1000)
in	*logical, optional :: maxit*	Maximization (.True.) or minimization (.False.) of function (default: .False.)
in	*logical, optional :: mask(size(pini))*	parameter to be optimized (true or false) (default: .True.)
in	*character(len=∗) , optional :: tmp_file*	file with temporal output
out	*real(dp), optional :: funcbest*	the best value of the function.
out	*real(dp), optional, allocatable :: history(:)*	the history of best function values,

Returns

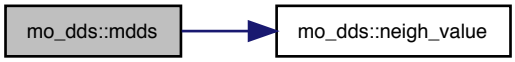
real(dp) :: MDDS — The parameters of the point which is estimated to minimize the function.

Author

Written Matthias Cuntz and Juliane Mai

References mo_kind::dp, mo_kind::i4, mo_kind::i8, and neigh_value().

Here is the call graph for this function:



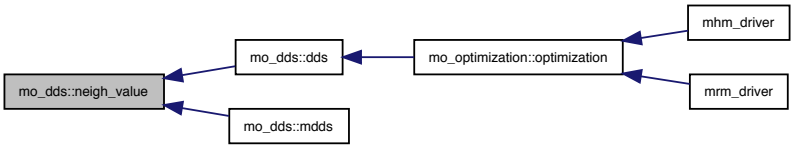
16.18.2.3 neigh_value()

```
subroutine mo_dds::neigh_value (
             real(dp), intent(in) x_cur,
             real(dp), intent(in) x_min,
             real(dp), intent(in) x_max,
             real(dp), intent(in) r,
             real(dp), intent(out) new_value )
```

References mo_kind::dp, and mo_kind::i8.

Referenced by dds(), and mdds().

Here is the caller graph for this function:



16.19 mo_errormeasures Module Reference

Data Types

- interface bias
- interface kge

  Kling-Gupta-Efficiency measure.
- interface kgenocorr

  Kling-Gupta-Efficiency measure without correlation.
- interface lnnse
- interface mae

- interface mse
- interface nse
- interface rmse
- interface sae
- interface sse
- interface wnse

Functions/Subroutines

- real(sp) function bias_sp_1d (x, y, mask)
- real(dp) function bias_dp_1d (x, y, mask)
- real(sp) function bias_sp_2d (x, y, mask)
- real(dp) function bias_dp_2d (x, y, mask)
- real(sp) function bias_sp_3d (x, y, mask)
- real(dp) function bias_dp_3d (x, y, mask)
- real(sp) function kge_sp_1d (x, y, mask)
- real(sp) function kge_sp_2d (x, y, mask)
- real(sp) function kge_sp_3d (x, y, mask)
- real(dp) function kge_dp_1d (x, y, mask)
- real(dp) function kge_dp_2d (x, y, mask)
- real(dp) function kge_dp_3d (x, y, mask)
- real(sp) function kgenocorr_sp_1d (x, y, mask)
- real(sp) function kgenocorr_sp_2d (x, y, mask)
- real(sp) function kgenocorr_sp_3d (x, y, mask)
- real(dp) function kgenocorr_dp_1d (x, y, mask)
- real(dp) function kgenocorr_dp_2d (x, y, mask)
- real(dp) function kgenocorr_dp_3d (x, y, mask)
- real(sp) function lnnse_sp_1d (x, y, mask)
- real(dp) function lnnse_dp_1d (x, y, mask)
- real(sp) function lnnse_sp_2d (x, y, mask)
- real(dp) function lnnse_dp_2d (x, y, mask)
- real(sp) function lnnse_sp_3d (x, y, mask)
- real(dp) function lnnse_dp_3d (x, y, mask)
- real(sp) function mae_sp_1d (x, y, mask)
- real(dp) function mae_dp_1d (x, y, mask)
- real(sp) function mae_sp_2d (x, y, mask)
- real(dp) function mae_dp_2d (x, y, mask)
- real(sp) function mae_sp_3d (x, y, mask)
- real(dp) function mae_dp_3d (x, y, mask)
- real(sp) function mse_sp_1d (x, y, mask)
- real(dp) function mse_dp_1d (x, y, mask)
- real(sp) function mse_sp_2d (x, y, mask)
- real(dp) function mse_dp_2d (x, y, mask)
- real(sp) function mse_sp_3d (x, y, mask)
- real(dp) function mse_dp_3d (x, y, mask)
- real(sp) function nse_sp_1d (x, y, mask)
- real(dp) function nse_dp_1d (x, y, mask)
- real(sp) function nse_sp_2d (x, y, mask)
- real(dp) function nse_dp_2d (x, y, mask)
- real(sp) function nse_sp_3d (x, y, mask)
- real(dp) function nse_dp_3d (x, y, mask)
- real(sp) function sae_sp_1d (x, y, mask)
- real(dp) function sae_dp_1d (x, y, mask)
- real(sp) function sae_sp_2d (x, y, mask)

- real(dp) function sae_dp_2d (x, y, mask)
- real(sp) function sae_sp_3d (x, y, mask)
- real(dp) function sae_dp_3d (x, y, mask)
- real(sp) function sse_sp_1d (x, y, mask)
- real(dp) function sse_dp_1d (x, y, mask)
- real(sp) function sse_sp_2d (x, y, mask)
- real(dp) function sse_dp_2d (x, y, mask)
- real(sp) function sse_sp_3d (x, y, mask)
- real(dp) function sse_dp_3d (x, y, mask)
- real(sp) function rmse_sp_1d (x, y, mask)
- real(dp) function rmse_dp_1d (x, y, mask)
- real(sp) function rmse_sp_2d (x, y, mask)
- real(dp) function rmse_dp_2d (x, y, mask)
- real(sp) function rmse_sp_3d (x, y, mask)
- real(dp) function rmse_dp_3d (x, y, mask)
- real(sp) function wnse_sp_1d (x, y, mask)
- real(dp) function wnse_dp_1d (x, y, mask)
- real(sp) function wnse_sp_2d (x, y, mask)
- real(dp) function wnse_dp_2d (x, y, mask)
- real(sp) function wnse_sp_3d (x, y, mask)
- real(dp) function wnse_dp_3d (x, y, mask)

16.19.1 Function/Subroutine Documentation

16.19.1.1 bias_dp_1d()

```
real(dp) function mo_errormeasures::bias_dp_1d (
             real(dp), dimension(:), intent(in) x,
             real(dp), dimension(:), intent(in) y,
             logical, dimension(:), intent(in), optional mask )
```

16.19.1.2 bias_dp_2d()

```
real(dp) function mo_errormeasures::bias_dp_2d (
             real(dp), dimension(:, :), intent(in) x,
             real(dp), dimension(:, :), intent(in) y,
             logical, dimension(:, :), intent(in), optional mask )
```

16.19.1.3 bias_dp_3d()

```
real(dp) function mo_errormeasures::bias_dp_3d (
             real(dp), dimension(:, :, :), intent(in) x,
             real(dp), dimension(:, :, :), intent(in) y,
             logical, dimension(:, :, :), intent(in), optional mask )
```

16.19.1.4 bias_sp_1d()

```
real(sp) function mo_errormeasures::bias_sp_1d (
             real(sp), dimension(:), intent(in) x,
             real(sp), dimension(:), intent(in) y,
             logical, dimension(:), intent(in), optional mask )  [private]
```

16.19.1.5 bias_sp_2d()

```
real(sp) function mo_errormeasures::bias_sp_2d (
             real(sp), dimension(:, :), intent(in) x,
             real(sp), dimension(:, :), intent(in) y,
             logical, dimension(:, :), intent(in), optional mask )
```

16.19.1.6 bias_sp_3d()

```
real(sp) function mo_errormeasures::bias_sp_3d (
             real(sp), dimension(:, :, :), intent(in) x,
             real(sp), dimension(:, :, :), intent(in) y,
             logical, dimension(:, :, :), intent(in), optional mask )
```

16.19.1.7 kge_dp_1d()

```
real(dp) function mo_errormeasures::kge_dp_1d (
             real(dp), dimension(:), intent(in) x,
             real(dp), dimension(:), intent(in) y,
             logical, dimension(:), intent(in), optional mask )
```

16.19.1.8 kge_dp_2d()

```
real(dp) function mo_errormeasures::kge_dp_2d (
             real(dp), dimension(:, :), intent(in) x,
             real(dp), dimension(:, :), intent(in) y,
             logical, dimension(:, :), intent(in), optional mask )
```

16.19.1.9 kge_dp_3d()

```
real(dp) function mo_errormeasures::kge_dp_3d (
             real(dp), dimension(:, :, :), intent(in) x,
             real(dp), dimension(:, :, :), intent(in) y,
             logical, dimension(:, :, :), intent(in), optional mask )
```

16.19.1.10 kge_sp_1d()

```
real(sp) function mo_errormeasures::kge_sp_1d (
```

```
            real(sp), dimension(:), intent(in) x,
            real(sp), dimension(:), intent(in) y,
            logical, dimension(:), intent(in), optional mask )
```

16.19.1.11 kge_sp_2d()

```
real(sp) function mo_errormeasures::kge_sp_2d (
            real(sp), dimension(:, :), intent(in) x,
            real(sp), dimension(:, :), intent(in) y,
            logical, dimension(:, :), intent(in), optional mask )
```

16.19.1.12 kge_sp_3d()

```
real(sp) function mo_errormeasures::kge_sp_3d (
            real(sp), dimension(:, :, :), intent(in) x,
            real(sp), dimension(:, :, :), intent(in) y,
            logical, dimension(:, :, :), intent(in), optional mask )
```

16.19.1.13 kgenocorr_dp_1d()

```
real(dp) function mo_errormeasures::kgenocorr_dp_1d (
            real(dp), dimension(:), intent(in) x,
            real(dp), dimension(:), intent(in) y,
            logical, dimension(:), intent(in), optional mask )
```

16.19.1.14 kgenocorr_dp_2d()

```
real(dp) function mo_errormeasures::kgenocorr_dp_2d (
            real(dp), dimension(:, :), intent(in) x,
            real(dp), dimension(:, :), intent(in) y,
            logical, dimension(:, :), intent(in), optional mask )
```

16.19.1.15 kgenocorr_dp_3d()

```
real(dp) function mo_errormeasures::kgenocorr_dp_3d (
            real(dp), dimension(:, :, :), intent(in) x,
            real(dp), dimension(:, :, :), intent(in) y,
            logical, dimension(:, :, :), intent(in), optional mask )
```

16.19.1.16 kgenocorr_sp_1d()

```
real(sp) function mo_errormeasures::kgenocorr_sp_1d (
            real(sp), dimension(:), intent(in) x,
            real(sp), dimension(:), intent(in) y,
            logical, dimension(:), intent(in), optional mask )
```

16.19.1.17 kgenocorr_sp_2d()

```
real(sp) function mo_errormeasures::kgenocorr_sp_2d (
             real(sp), dimension(:, :), intent(in) x,
             real(sp), dimension(:, :), intent(in) y,
             logical, dimension(:, :), intent(in), optional mask )
```

16.19.1.18 kgenocorr_sp_3d()

```
real(sp) function mo_errormeasures::kgenocorr_sp_3d (
             real(sp), dimension(:, :, :), intent(in) x,
             real(sp), dimension(:, :, :), intent(in) y,
             logical, dimension(:, :, :), intent(in), optional mask )
```

16.19.1.19 lnnse_dp_1d()

```
real(dp) function mo_errormeasures::lnnse_dp_1d (
             real(dp), dimension(:), intent(in) x,
             real(dp), dimension(:), intent(in) y,
             logical, dimension(:), intent(inout), optional mask )
```

16.19.1.20 lnnse_dp_2d()

```
real(dp) function mo_errormeasures::lnnse_dp_2d (
             real(dp), dimension(:, :), intent(in) x,
             real(dp), dimension(:, :), intent(in) y,
             logical, dimension(:, :), intent(inout), optional mask )
```

16.19.1.21 lnnse_dp_3d()

```
real(dp) function mo_errormeasures::lnnse_dp_3d (
             real(dp), dimension(:, :, :), intent(in) x,
             real(dp), dimension(:, :, :), intent(in) y,
             logical, dimension(:, :, :), intent(inout), optional mask )
```

16.19.1.22 lnnse_sp_1d()

```
real(sp) function mo_errormeasures::lnnse_sp_1d (
             real(sp), dimension(:), intent(in) x,
             real(sp), dimension(:), intent(in) y,
             logical, dimension(:), intent(inout), optional mask )
```

16.19.1.23 lnnse_sp_2d()

```
real(sp) function mo_errormeasures::lnnse_sp_2d (
             real(sp), dimension(:, :), intent(in) x,
             real(sp), dimension(:, :), intent(in) y,
             logical, dimension(:, :), intent(inout), optional mask )
```

16.19.1.24 lnnse_sp_3d()

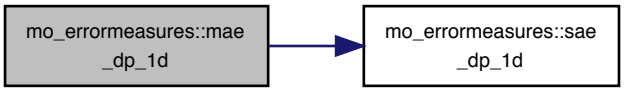
```
real(sp) function mo_errormeasures::lnnse_sp_3d (
             real(sp), dimension(:, :, :), intent(in) x,
             real(sp), dimension(:, :, :), intent(in) y,
             logical, dimension(:, :, :), intent(inout), optional mask )
```

16.19.1.25 mae_dp_1d()

```
real(dp) function mo_errormeasures::mae_dp_1d (
             real(dp), dimension(:), intent(in) x,
             real(dp), dimension(:), intent(in) y,
             logical, dimension(:), intent(in), optional mask )  [private]
```

References sae_dp_1d().

Here is the call graph for this function:

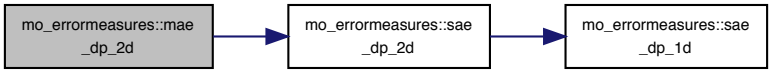


16.19.1.26 mae_dp_2d()

```
real(dp) function mo_errormeasures::mae_dp_2d (
             real(dp), dimension(:, :), intent(in) x,
             real(dp), dimension(:, :), intent(in) y,
             logical, dimension(:, :), intent(in), optional mask )  [private]
```

References sae_dp_2d().

Here is the call graph for this function:

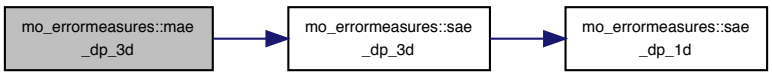


16.19.1.27 mae_dp_3d()

```
real(dp) function mo_errormeasures::mae_dp_3d (
            real(dp), dimension(:, :, :), intent(in) x,
            real(dp), dimension(:, :, :), intent(in) y,
            logical, dimension(:, :, :), intent(in), optional mask )  [private]
```

References sae_dp_3d().

Here is the call graph for this function:

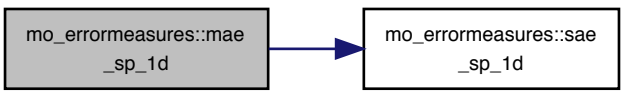


16.19.1.28 mae_sp_1d()

```
real(sp) function mo_errormeasures::mae_sp_1d (
            real(sp), dimension(:), intent(in) x,
            real(sp), dimension(:), intent(in) y,
            logical, dimension(:), intent(in), optional mask )
```

References sae_sp_1d().

Here is the call graph for this function:

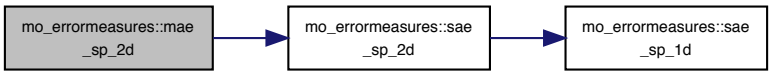


16.19.1.29 mae_sp_2d()

```
real(sp) function mo_errormeasures::mae_sp_2d (
            real(sp), dimension(:, :), intent(in) x,
            real(sp), dimension(:, :), intent(in) y,
            logical, dimension(:, :), intent(in), optional mask )  [private]
```

References sae_sp_2d().

Here is the call graph for this function:

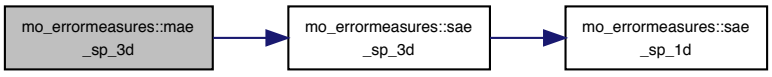


16.19.1.30 mae_sp_3d()

```
real(sp) function mo_errormeasures::mae_sp_3d (
            real(sp), dimension(:, :, :), intent(in) x,
            real(sp), dimension(:, :, :), intent(in) y,
            logical, dimension(:, :, :), intent(in), optional mask )  [private]
```

References sae_sp_3d().

Here is the call graph for this function:



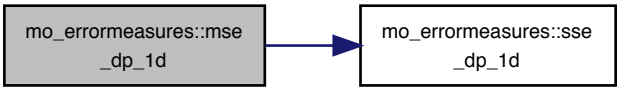
16.19.1.31 mse_dp_1d()

```
real(dp) function mo_errormeasures::mse_dp_1d (
            real(dp), dimension(:), intent(in) x,
            real(dp), dimension(:), intent(in) y,
            logical, dimension(:), intent(in), optional mask )  [private]
```

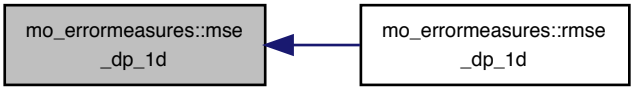
References sse_dp_1d().

Referenced by rmse_dp_1d().

Here is the call graph for this function:



Here is the caller graph for this function:



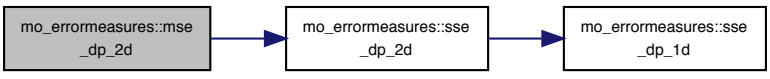
16.19.1.32 mse_dp_2d()

```
real(dp) function mo_errormeasures::mse_dp_2d (
             real(dp), dimension(:, :), intent(in) x,
             real(dp), dimension(:, :), intent(in) y,
             logical, dimension(:, :), intent(in), optional mask )  [private]
```

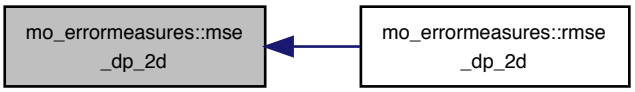
References sse_dp_2d().

Referenced by rmse_dp_2d().

Here is the call graph for this function:



Here is the caller graph for this function:



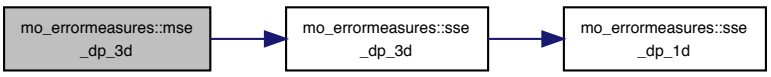
16.19.1.33 mse_dp_3d()

```
real(dp) function mo_errormeasures::mse_dp_3d (
            real(dp), dimension(:, :, :), intent(in) x,
            real(dp), dimension(:, :, :), intent(in) y,
            logical, dimension(:, :, :), intent(in), optional mask )  [private]
```

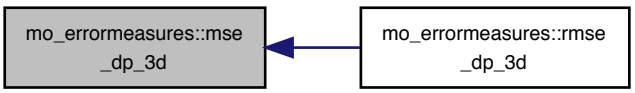
References sse_dp_3d().

Referenced by rmse_dp_3d().

Here is the call graph for this function:



Here is the caller graph for this function:



16.19.1.34 mse_sp_1d()

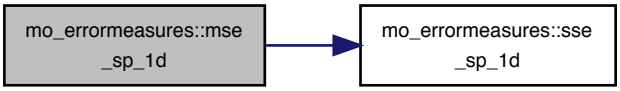
```
real(sp) function mo_errormeasures::mse_sp_1d (
            real(sp), dimension(:), intent(in) x,
```

```
            real(sp), dimension(:), intent(in) y,
            logical, dimension(:), intent(in), optional mask )  [private]
```

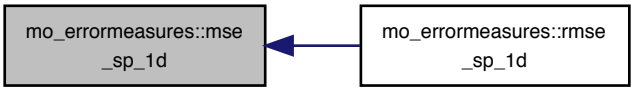
References sse_sp_1d().

Referenced by rmse_sp_1d().

Here is the call graph for this function:



Here is the caller graph for this function:



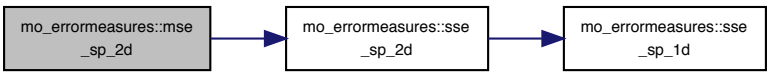
16.19.1.35 mse_sp_2d()

```
real(sp) function mo_errormeasures::mse_sp_2d (
            real(sp), dimension(:, :), intent(in) x,
            real(sp), dimension(:, :), intent(in) y,
            logical, dimension(:, :), intent(in), optional mask )  [private]
```

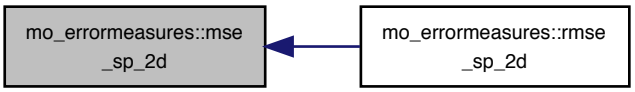
References sse_sp_2d().

Referenced by rmse_sp_2d().

Here is the call graph for this function:



Here is the caller graph for this function:



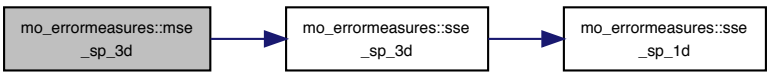
16.19.1.36 mse_sp_3d()

```
real(sp) function mo_errormeasures::mse_sp_3d (
            real(sp), dimension(:, :, :), intent(in) x,
            real(sp), dimension(:, :, :), intent(in) y,
            logical, dimension(:, :, :), intent(in), optional mask )  [private]
```

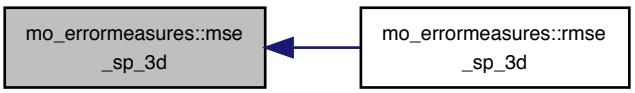
References sse_sp_3d().

Referenced by rmse_sp_3d().

Here is the call graph for this function:



Here is the caller graph for this function:



16.19.1.37 nse_dp_1d()

```
real(dp) function mo_errormeasures::nse_dp_1d (
            real(dp), dimension(:), intent(in) x,
```

```
            real(dp), dimension(:), intent(in) y,
            logical, dimension(:), intent(in), optional mask )
```

16.19.1.38 nse_dp_2d()

```
real(dp) function mo_errormeasures::nse_dp_2d (
            real(dp), dimension(:, :), intent(in) x,
            real(dp), dimension(:, :), intent(in) y,
            logical, dimension(:, :), intent(in), optional mask )
```

16.19.1.39 nse_dp_3d()

```
real(dp) function mo_errormeasures::nse_dp_3d (
            real(dp), dimension(:, :, :), intent(in) x,
            real(dp), dimension(:, :, :), intent(in) y,
            logical, dimension(:, :, :), intent(in), optional mask )
```

16.19.1.40 nse_sp_1d()

```
real(sp) function mo_errormeasures::nse_sp_1d (
            real(sp), dimension(:), intent(in) x,
            real(sp), dimension(:), intent(in) y,
            logical, dimension(:), intent(in), optional mask )  [private]
```

16.19.1.41 nse_sp_2d()

```
real(sp) function mo_errormeasures::nse_sp_2d (
            real(sp), dimension(:, :), intent(in) x,
            real(sp), dimension(:, :), intent(in) y,
            logical, dimension(:, :), intent(in), optional mask )
```

16.19.1.42 nse_sp_3d()

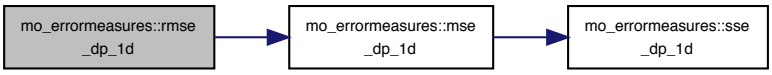
```
real(sp) function mo_errormeasures::nse_sp_3d (
            real(sp), dimension(:, :, :), intent(in) x,
            real(sp), dimension(:, :, :), intent(in) y,
            logical, dimension(:, :, :), intent(in), optional mask )
```

16.19.1.43 rmse_dp_1d()

```
real(dp) function mo_errormeasures::rmse_dp_1d (
            real(dp), dimension(:), intent(in) x,
            real(dp), dimension(:), intent(in) y,
            logical, dimension(:), intent(in), optional mask )  [private]
```

References mse_dp_1d().

Here is the call graph for this function:

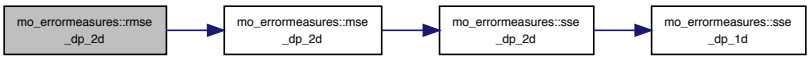


16.19.1.44 rmse_dp_2d()

```
real(dp) function mo_errormeasures::rmse_dp_2d (
            real(dp), dimension(:, :), intent(in) x,
            real(dp), dimension(:, :), intent(in) y,
            logical, dimension(:, :), intent(in), optional mask )  [private]
```

References mse_dp_2d().

Here is the call graph for this function:

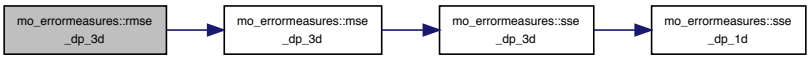


16.19.1.45 rmse_dp_3d()

```
real(dp) function mo_errormeasures::rmse_dp_3d (
            real(dp), dimension(:, :, :), intent(in) x,
            real(dp), dimension(:, :, :), intent(in) y,
            logical, dimension(:, :, :), intent(in), optional mask )  [private]
```

References mse_dp_3d().

Here is the call graph for this function:

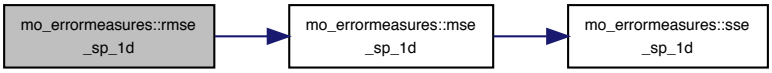


16.19.1.46 rmse_sp_1d()

```
real(sp) function mo_errormeasures::rmse_sp_1d (
            real(sp), dimension(:), intent(in) x,
            real(sp), dimension(:), intent(in) y,
            logical, dimension(:), intent(in), optional mask )  [private]
```

References mse_sp_1d().

Here is the call graph for this function:

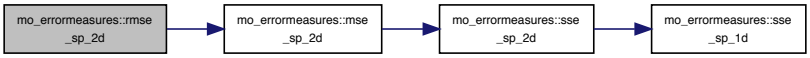


16.19.1.47 rmse_sp_2d()

```
real(sp) function mo_errormeasures::rmse_sp_2d (
            real(sp), dimension(:, :), intent(in) x,
            real(sp), dimension(:, :), intent(in) y,
            logical, dimension(:, :), intent(in), optional mask )  [private]
```

References mse_sp_2d().

Here is the call graph for this function:

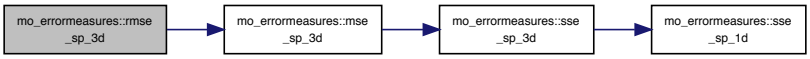


16.19.1.48 rmse_sp_3d()

```
real(sp) function mo_errormeasures::rmse_sp_3d (
            real(sp), dimension(:, :, :), intent(in) x,
            real(sp), dimension(:, :, :), intent(in) y,
            logical, dimension(:, :, :), intent(in), optional mask )  [private]
```

References mse_sp_3d().

Here is the call graph for this function:

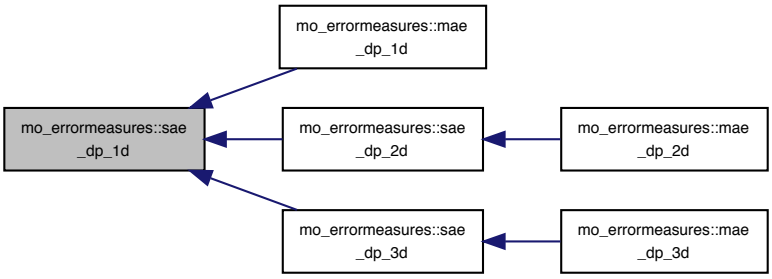


16.19.1.49 sae_dp_1d()

```
real(dp) function mo_errormeasures::sae_dp_1d (
            real(dp), dimension(:), intent(in) x,
            real(dp), dimension(:), intent(in) y,
            logical, dimension(:), intent(in), optional mask )  [private]
```

Referenced by mae_dp_1d(), sae_dp_2d(), and sae_dp_3d().

Here is the caller graph for this function:



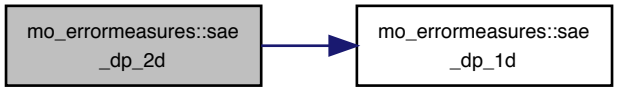
16.19.1.50 sae_dp_2d()

```
real(dp) function mo_errormeasures::sae_dp_2d (
            real(dp), dimension(:, :), intent(in) x,
            real(dp), dimension(:, :), intent(in) y,
            logical, dimension(:, :), intent(in), optional mask )  [private]
```

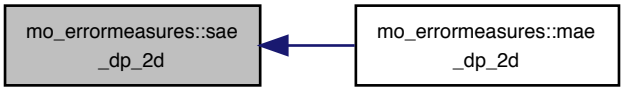
References sae_dp_1d().

Referenced by mae_dp_2d().

Here is the call graph for this function:



Here is the caller graph for this function:



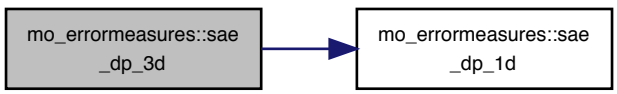
16.19.1.51 sae_dp_3d()

```
real(dp) function mo_errormeasures::sae_dp_3d (
            real(dp), dimension(:, :, :), intent(in) x,
            real(dp), dimension(:, :, :), intent(in) y,
            logical, dimension(:, :, :), intent(in), optional mask )  [private]
```

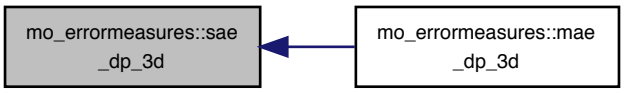
References sae_dp_1d().

Referenced by mae_dp_3d().

Here is the call graph for this function:



Here is the caller graph for this function:

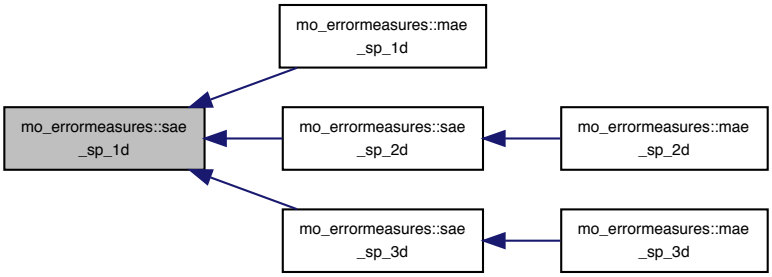


16.19.1.52 sae_sp_1d()

```
real(sp) function mo_errormeasures::sae_sp_1d (
            real(sp), dimension(:), intent(in) x,
            real(sp), dimension(:), intent(in) y,
            logical, dimension(:), intent(in), optional mask )
```

Referenced by mae_sp_1d(), sae_sp_2d(), and sae_sp_3d().

Here is the caller graph for this function:



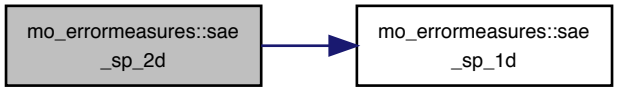
16.19.1.53 sae_sp_2d()

```
real(sp) function mo_errormeasures::sae_sp_2d (
            real(sp), dimension(:, :), intent(in) x,
            real(sp), dimension(:, :), intent(in) y,
            logical, dimension(:, :), intent(in), optional mask )  [private]
```

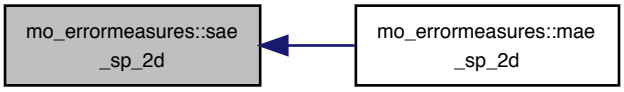
References sae_sp_1d().

Referenced by mae_sp_2d().

Here is the call graph for this function:



Here is the caller graph for this function:



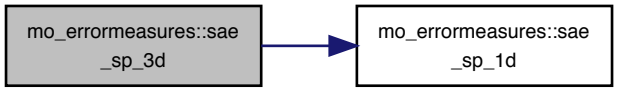
16.19.1.54 sae_sp_3d()

```
real(sp) function mo_errormeasures::sae_sp_3d (
             real(sp), dimension(:, :, :), intent(in) x,
             real(sp), dimension(:, :, :), intent(in) y,
             logical, dimension(:, :, :), intent(in), optional mask )  [private]
```

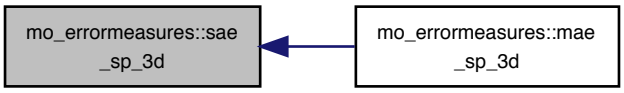
References sae_sp_1d().

Referenced by mae_sp_3d().

Here is the call graph for this function:



Here is the caller graph for this function:

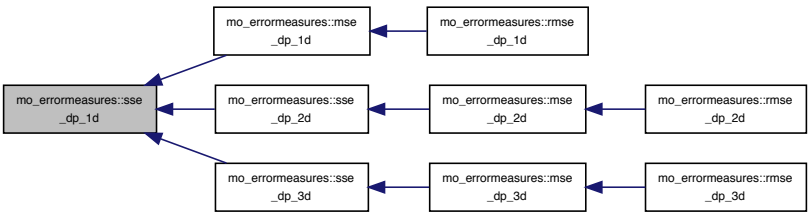


16.19.1.55 sse_dp_1d()

```
real(dp) function mo_errormeasures::sse_dp_1d (
            real(dp), dimension(:), intent(in) x,
            real(dp), dimension(:), intent(in) y,
            logical, dimension(:), intent(in), optional mask )  [private]
```

Referenced by mse_dp_1d(), sse_dp_2d(), and sse_dp_3d().

Here is the caller graph for this function:



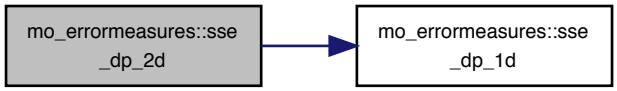
16.19.1.56 sse_dp_2d()

```
real(dp) function mo_errormeasures::sse_dp_2d (
            real(dp), dimension(:, :), intent(in) x,
            real(dp), dimension(:, :), intent(in) y,
            logical, dimension(:, :), intent(in), optional mask )  [private]
```

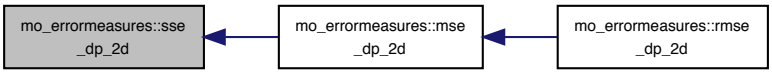
References sse_dp_1d().

Referenced by mse_dp_2d().

Here is the call graph for this function:



Here is the caller graph for this function:



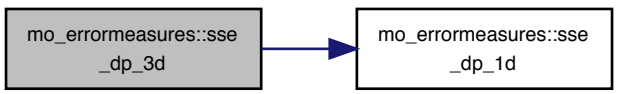
16.19.1.57 sse_dp_3d()

```
real(dp) function mo_errormeasures::sse_dp_3d (
             real(dp), dimension(:, :, :), intent(in) x,
             real(dp), dimension(:, :, :), intent(in) y,
             logical, dimension(:, :, :), intent(in), optional mask )  [private]
```

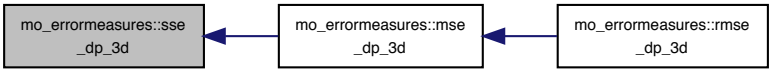
References sse_dp_1d().

Referenced by mse_dp_3d().

Here is the call graph for this function:



Here is the caller graph for this function:

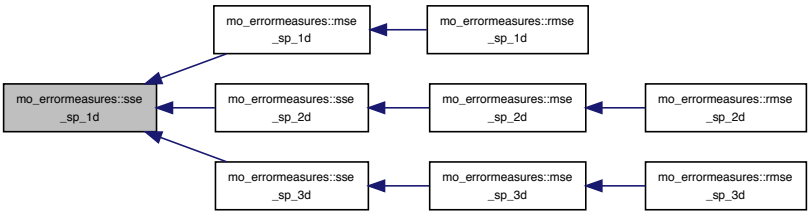


16.19.1.58 sse_sp_1d()

```
real(sp) function mo_errormeasures::sse_sp_1d (
            real(sp), dimension(:), intent(in) x,
            real(sp), dimension(:), intent(in) y,
            logical, dimension(:), intent(in), optional mask )  [private]
```

Referenced by mse_sp_1d(), sse_sp_2d(), and sse_sp_3d().

Here is the caller graph for this function:



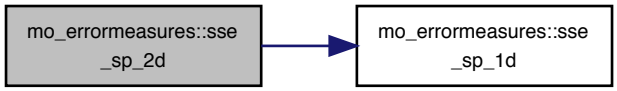
16.19.1.59 sse_sp_2d()

```
real(sp) function mo_errormeasures::sse_sp_2d (
            real(sp), dimension(:, :), intent(in) x,
            real(sp), dimension(:, :), intent(in) y,
            logical, dimension(:, :), intent(in), optional mask )  [private]
```

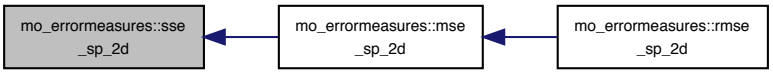
References sse_sp_1d().

Referenced by mse_sp_2d().

Here is the call graph for this function:



Here is the caller graph for this function:



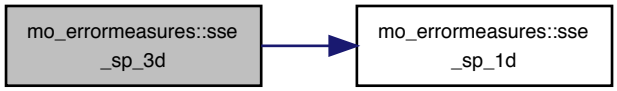
16.19.1.60 sse_sp_3d()

```
real(sp) function mo_errormeasures::sse_sp_3d (
             real(sp), dimension(:, :, :), intent(in) x,
             real(sp), dimension(:, :, :), intent(in) y,
             logical, dimension(:, :, :), intent(in), optional mask )  [private]
```

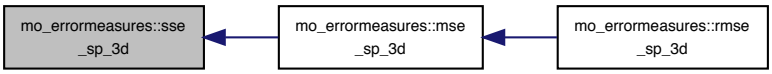
References sse_sp_1d().

Referenced by mse_sp_3d().

Here is the call graph for this function:



Here is the caller graph for this function:



16.19.1.61 wnse_dp_1d()

```
real(dp) function mo_errormeasures::wnse_dp_1d (
            real(dp), dimension(:), intent(in) x,
            real(dp), dimension(:), intent(in) y,
            logical, dimension(:), intent(in), optional mask )
```

16.19.1.62 wnse_dp_2d()

```
real(dp) function mo_errormeasures::wnse_dp_2d (
            real(dp), dimension(:,:), intent(in) x,
            real(dp), dimension(:,:), intent(in) y,
            logical, dimension(:,:), intent(in), optional mask )
```

16.19.1.63 wnse_dp_3d()

```
real(dp) function mo_errormeasures::wnse_dp_3d (
            real(dp), dimension(:,:,:), intent(in) x,
            real(dp), dimension(:,:,:), intent(in) y,
            logical, dimension(:,:,:), intent(in), optional mask )
```

16.19.1.64 wnse_sp_1d()

```
real(sp) function mo_errormeasures::wnse_sp_1d (
            real(sp), dimension(:), intent(in) x,
            real(sp), dimension(:), intent(in) y,
            logical, dimension(:), intent(in), optional mask )  [private]
```

16.19.1.65 wnse_sp_2d()

```
real(sp) function mo_errormeasures::wnse_sp_2d (
            real(sp), dimension(:,:), intent(in) x,
            real(sp), dimension(:,:), intent(in) y,
            logical, dimension(:,:), intent(in), optional mask )
```

16.19.1.66 wnse_sp_3d()

```
real(sp) function mo_errormeasures::wnse_sp_3d (
             real(sp), dimension(:,:,:), intent(in) x,
             real(sp), dimension(:,:,:), intent(in) y,
             logical, dimension(:,:,:), intent(in), optional mask )
```

16.20 mo_file Module Reference

Provides file names and units for mHM.

Variables

- character(len=∗), parameter version = '5.10'

  Current mHM model version.
- character(len=∗), parameter version_date = 'June 2019'

  Time of current mHM model version release.
- character(len=∗), parameter file_main = 'mhm_driver.f90'

  Driver file.
- character(len=∗), parameter file_namelist_mhm = 'mhm.nml'

  Namelist file name.
- integer, parameter unamelist_mhm = 30

  Unit for namelist.
- character(len=∗), parameter file_namelist_mhm_param = 'mhm_parameter.nml'

  Parameter namelists file name.
- integer, parameter unamelist_mhm_param = 31

  Unit for namelist.
- character(len=∗), parameter file_defoutput = 'mhm_outputs.nml'

  file defining mHM's outputs
- integer, parameter udefoutput = 67

  Unit for file defining mHM's outputs.
- integer, parameter utws = 77

  unit for tws time series

16.20.1 Detailed Description

Provides file names and units for mHM.

Provides all filenames as well as all units used for the hydrologic model mHM.

Authors

Matthias Cuntz

Date

Jan 2012

16.20.2 Variable Documentation

16.20.2.1 file_defoutput

```
character(len = *), parameter mo_file::file_defoutput = 'mhm_outputs.nml'
```

file defining mHM's outputs

Referenced by mo_mhm_read_config::mhm_read_config().

16.20.2.2 file_main

```
character(len = *), parameter mo_file::file_main = 'mhm_driver.f90'
```

Driver file.

Referenced by mhm_driver().

16.20.2.3 file_namelist_mhm

```
character(len = *), parameter mo_file::file_namelist_mhm = 'mhm.nml'
```

Namelist file name.

16.20.2.4 file_namelist_mhm_param

```
character(len = *), parameter mo_file::file_namelist_mhm_param = 'mhm_parameter.nml'
```

Parameter namelists file name.

Referenced by mo_startup::constants_init().

16.20.2.5 udefoutput

```
integer, parameter mo_file::udefoutput = 67
```

Unit for file defining mHM's outputs.

Referenced by mo_mhm_read_config::mhm_read_config().

16.20.2.6 unamelist_mhm

```
integer, parameter mo_file::unamelist_mhm = 30
```

Unit for namelist.

16.20.2.7 unamelist_mhm_param

```
integer, parameter mo_file::unamelist_mhm_param = 31
```

Unit for namelist.

16.20.2.8 utws

```
integer, parameter mo_file::utws = 77
```

unit for tws time series

Referenced by mo_read_optional_data::read_basin_avg_tws().

16.20.2.9 version

```
character(len = *), parameter mo_file::version = '5.10'
```

Current mHM model version.

Referenced by mo_mrm_river_head::create_output(), mo_write_fluxes_states::createoutputfile(), mhm_driver(), and mo_write_ascii::write_configfile().

16.20.2.10 version_date

```
character(len = *), parameter mo_file::version_date = 'June 2019'
```

Time of current mHM model version release.

Referenced by mhm_driver().

16.21 mo_finish Module Reference

Finish a program gracefully.

Functions/Subroutines

- subroutine, public finish (name, text, unit)

  Finish a program gracefully.

16.21.1 Detailed Description

Finish a program gracefully.

This module supplies a routine that writes out final comments and then stops.

Authors

Original of Echam5, (C) MPI-MET, Hamburg, Germany.
Modified Matthias Cuntz

Date

Jan 2011

16.21.2 Function/Subroutine Documentation

16.21.2.1 finish()

```
subroutine, public mo_finish::finish (
             character(len = *), intent(in) name,
             character(len = *), intent(in), optional text,
             integer, intent(in), optional unit )
```

Finish a program gracefully.

Stop a program but writing out a message first that is separated from earlier output by ---------- (i.e. the separator of mo_string_utils)

Parameters

in	*character(len=*) :: name*	First string separated from otional second by :
in	*character(len=*), optional :: text*	Second string separated by :
in	*integer, optional :: unit*	File unit for write (default: *)

Author

Written, Matthias Cuntz - modified from Echam5, (C) MPI-MET, Hamburg, Germany

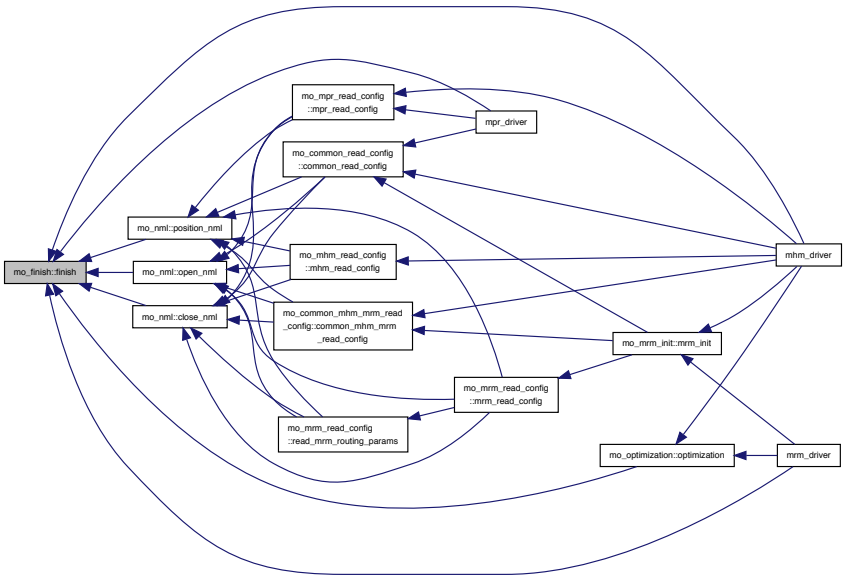
Date

Dec 2011

References mo_string_utils::separator.

Referenced by mo_nml::close_nml(), mhm_driver(), mpr_driver(), mrm_driver(), mo_nml::open_nml(), mo_optimization::optimization(), and mo_nml::position_nml().

Here is the caller graph for this function:



16.22 mo_global_variables Module Reference

Global variables ONLY used in reading, writing and startup.

Data Types

- type twsstructure

Variables

- integer(i4) timestep_model_outputs
- logical, dimension(noutflxstate) outputflxstate
- integer(i4), dimension(:), allocatable, public timestep_model_inputs
- logical, public read_meteo_weights
- character(256), public inputformat_meteo_forcings
- integer(i4), public timestep_sm_input
- integer(i4), public timestep_neutrons_input
- integer(i4), public timestep_et_input
- character(256), dimension(:), allocatable, public dirprecipitation
- character(256), dimension(:), allocatable, public dirtemperature
- character(256), dimension(:), allocatable, public dirmintemperature
- character(256), dimension(:), allocatable, public dirmaxtemperature
- character(256), dimension(:), allocatable, public dirnetradiation
- character(256), dimension(:), allocatable, public dirabsvappressure
- character(256), dimension(:), allocatable, public dirwindspeed
- character(256), dimension(:), allocatable, public dirreferenceet
- character(256), dimension(:), allocatable, public dirsoil_moisture
- character(256), dimension(:), allocatable, public filetws
- character(256), dimension(:), allocatable, public dirneutrons
- character(256), dimension(:), allocatable, public direvapotranspiration
- integer(i4), parameter, public routingstates = 2
- type(twsstructure), public basin_avg_tws_obs
- real(dp), dimension(:, :), allocatable, public basin_avg_tws_sim
- integer(i4), public nmeasperday_tws
- type(grid), dimension(:), allocatable, public level2
- real(dp), dimension(:, :, :), allocatable, public l1_temp_weights
- real(dp), dimension(:, :, :), allocatable, public l1_pet_weights
- real(dp), dimension(:, :, :), allocatable, public l1_pre_weights
- real(dp), dimension(:, :), allocatable, public l1_pre
- real(dp), dimension(:, :), allocatable, public l1_temp
- real(dp), dimension(:, :), allocatable, public l1_pet
- real(dp), dimension(:, :), allocatable, public l1_tmin
- real(dp), dimension(:, :), allocatable, public l1_tmax
- real(dp), dimension(:, :), allocatable, public l1_netrad
- real(dp), dimension(:, :), allocatable, public l1_absvappress
- real(dp), dimension(:, :), allocatable, public l1_windspeed
- real(dp), dimension(:, :), allocatable, public l1_sm
- logical, dimension(:, :), allocatable, public l1_sm_mask
- integer(i4) ntimesteps_l1_sm
- integer(i4) nsoilhorizons_sm_input
- real(dp), dimension(:, :), allocatable, public l1_neutronsdata
- logical, dimension(:, :), allocatable, public l1_neutronsdata_mask
- integer(i4) ntimesteps_l1_neutrons

- real(dp), dimension(:, :), allocatable, public l1_et
- logical, dimension(:, :), allocatable, public l1_et_mask
- integer(i4) ntimesteps_l1_et
- real(dp), dimension(:), allocatable, public l1_inter
- real(dp), dimension(:), allocatable, public l1_snowpack
- real(dp), dimension(:), allocatable, public l1_sealstw
- real(dp), dimension(:, :), allocatable, public l1_soilmoist
- real(dp), dimension(:), allocatable, public l1_unsatstw
- real(dp), dimension(:), allocatable, public l1_satstw
- real(dp), dimension(:), allocatable, public l1_neutrons
- real(dp), dimension(:), allocatable, public l1_pet_calc
- real(dp), dimension(:, :), allocatable, public l1_aetsoil
- real(dp), dimension(:), allocatable, public l1_aetcanopy
- real(dp), dimension(:), allocatable, public l1_aetsealed
- real(dp), dimension(:), allocatable, public l1_baseflow
- real(dp), dimension(:, :), allocatable, public l1_infilsoil
- real(dp), dimension(:), allocatable, public l1_fastrunoff
- real(dp), dimension(:), allocatable, public l1_melt
- real(dp), dimension(:), allocatable, public l1_percol
- real(dp), dimension(:), allocatable, public l1_preeffect
- real(dp), dimension(:), allocatable, public l1_rain
- real(dp), dimension(:), allocatable, public l1_runoffseal
- real(dp), dimension(:), allocatable, public l1_slowrunoff
- real(dp), dimension(:), allocatable, public l1_snow
- real(dp), dimension(:), allocatable, public l1_throughfall
- real(dp), dimension(:), allocatable, public l1_total_runoff
- real(dp), dimension(int(yearmonths, i4)), public evap_coeff
- real(dp), dimension(int(yearmonths, i4)), public fday_prec
- real(dp), dimension(int(yearmonths, i4)), public fnight_prec
- real(dp), dimension(int(yearmonths, i4)), public fday_pet
- real(dp), dimension(int(yearmonths, i4)), public fnight_pet
- real(dp), dimension(int(yearmonths, i4)), public fday_temp
- real(dp), dimension(int(yearmonths, i4)), public fnight_temp
- real(dp), dimension(:), allocatable, public neutron_integral_afast

16.22.1 Detailed Description

Global variables ONLY used in reading, writing and startup.

TODO: add description

Authors

Luis Samaniego

Date

Dec 2012

16.22.2 Variable Documentation

16.22.2.1 basin_avg_tws_obs

```
type(twsstructure), public mo_global_variables::basin_avg_tws_obs
```

Referenced by mo_objective_function::extract_basin_avg_tws(), mo_mhm_read_config::mhm_read_config(), and mo_read_optional_data::read_basin_avg_tws().

16.22.2.2 basin_avg_tws_sim

```
real(dp), dimension(:, :), allocatable, public mo_global_variables::basin_avg_tws_sim
```

Referenced by mo_mhm_eval::mhm_eval(), and mo_read_optional_data::read_basin_avg_tws().

16.22.2.3 dirabsvappressure

```
character(256), dimension(:), allocatable, public mo_global_variables::dirabsvappressure
```

Referenced by mo_mhm_read_config::mhm_read_config(), and mo_meteo_forcings::prepare_meteo_forcings_↩data().

16.22.2.4 direvapotranspiration

```
character(256), dimension(:), allocatable, public mo_global_variables::direvapotranspiration
```

Referenced by mo_mhm_read_config::mhm_read_config(), and mo_read_optional_data::read_evapotranspiration().

16.22.2.5 dirmaxtemperature

```
character(256), dimension(:), allocatable, public mo_global_variables::dirmaxtemperature
```

Referenced by mo_mhm_read_config::mhm_read_config(), and mo_meteo_forcings::prepare_meteo_forcings_↩data().

16.22.2.6 dirmintemperature

```
character(256), dimension(:), allocatable, public mo_global_variables::dirmintemperature
```

Referenced by mo_mhm_read_config::mhm_read_config(), and mo_meteo_forcings::prepare_meteo_forcings_↩data().

16.22.2.7 dirnetradiation

```
character(256), dimension(:), allocatable, public mo_global_variables::dirnetradiation
```

Referenced by mo_mhm_read_config::mhm_read_config(), and mo_meteo_forcings::prepare_meteo_forcings_↩data().

16.22.2.8 dirneutrons

```
character(256), dimension(:), allocatable, public mo_global_variables::dirneutrons
```

Referenced by mo_mhm_read_config::mhm_read_config(), and mo_read_optional_data::read_neutrons().

16.22.2.9 dirprecipitation

```
character(256), dimension(:), allocatable, public mo_global_variables::dirprecipitation
```

Referenced by mo_startup::l2_variable_init(), mhm_driver(), mo_mhm_read_config::mhm_read_config(), mo_↩ meteo_forcings::prepare_meteo_forcings_data(), and mo_write_ascii::write_configfile().

16.22.2.10 dirreferenceet

```
character(256), dimension(:), allocatable, public mo_global_variables::dirreferenceet
```

Referenced by mo_mhm_read_config::mhm_read_config(), mo_meteo_forcings::prepare_meteo_forcings_data(), and mo_write_ascii::write_configfile().

16.22.2.11 dirsoil_moisture

```
character(256), dimension(:), allocatable, public mo_global_variables::dirsoil_moisture
```

Referenced by mo_mhm_read_config::mhm_read_config(), and mo_read_optional_data::read_soil_moisture().

16.22.2.12 dirtemperature

```
character(256), dimension(:), allocatable, public mo_global_variables::dirtemperature
```

Referenced by mo_mhm_read_config::mhm_read_config(), mo_meteo_forcings::prepare_meteo_forcings_data(), and mo_write_ascii::write_configfile().

16.22.2.13 dirwindspeed

```
character(256), dimension(:), allocatable, public mo_global_variables::dirwindspeed
```

Referenced by mo_mhm_read_config::mhm_read_config(), and mo_meteo_forcings::prepare_meteo_forcings_↩ data().

16.22.2.14 evap_coeff

```
real(dp), dimension(int(yearmonths, i4)), public mo_global_variables::evap_coeff
```

Referenced by mo_mhm_eval::mhm_eval(), and mo_mhm_read_config::mhm_read_config().

16.22.2.15 fday_pet

```
real(dp), dimension(int(yearmonths, i4)), public mo_global_variables::fday_pet
```

Referenced by mo_mhm_eval::mhm_eval(), and mo_mhm_read_config::mhm_read_config().

16.22.2.16 fday_prec

```
real(dp), dimension(int(yearmonths, i4)), public mo_global_variables::fday_prec
```

Referenced by mo_mhm_eval::mhm_eval(), and mo_mhm_read_config::mhm_read_config().

16.22.2.17 fday_temp

```
real(dp), dimension(int(yearmonths, i4)), public mo_global_variables::fday_temp
```

Referenced by mo_mhm_eval::mhm_eval(), and mo_mhm_read_config::mhm_read_config().

16.22.2.18 filetws

```
character(256), dimension(:), allocatable, public mo_global_variables::filetws
```

Referenced by mo_mhm_read_config::mhm_read_config().

16.22.2.19 fnight_pet

```
real(dp), dimension(int(yearmonths, i4)), public mo_global_variables::fnight_pet
```

Referenced by mo_mhm_eval::mhm_eval(), and mo_mhm_read_config::mhm_read_config().

16.22.2.20 fnight_prec

```
real(dp), dimension(int(yearmonths, i4)), public mo_global_variables::fnight_prec
```

Referenced by mo_mhm_eval::mhm_eval(), and mo_mhm_read_config::mhm_read_config().

16.22.2.21 fnight_temp

```
real(dp), dimension(int(yearmonths, i4)), public mo_global_variables::fnight_temp
```

Referenced by mo_mhm_eval::mhm_eval(), and mo_mhm_read_config::mhm_read_config().

16.22.2.22 inputformat_meteo_forcings

```
character(256), public mo_global_variables::inputformat_meteo_forcings
```

Referenced by mo_mhm_read_config::mhm_read_config(), and mo_meteo_forcings::prepare_meteo_forcings_↩
data().

16.22.2.23 l1_absvappress

```
real(dp), dimension(:, :), allocatable, public mo_global_variables::l1_absvappress
```

Referenced by mo_mhm_eval::mhm_eval(), and mo_meteo_forcings::prepare_meteo_forcings_data().

16.22.2.24 l1_aetcanopy

```
real(dp), dimension(:), allocatable, public mo_global_variables::l1_aetcanopy
```

Referenced by mo_mhm_eval::mhm_eval(), mo_restart::read_restart_states(), mo_init_states::variables_alloc(), mo_init_states::variables_default_init(), and mo_restart::write_restart_files().

16.22.2.25 l1_aetsealed

```
real(dp), dimension(:), allocatable, public mo_global_variables::l1_aetsealed
```

Referenced by mo_mhm_eval::mhm_eval(), mo_restart::read_restart_states(), mo_init_states::variables_alloc(), mo_init_states::variables_default_init(), and mo_restart::write_restart_files().

16.22.2.26 l1_aetsoil

```
real(dp), dimension(:, :), allocatable, public mo_global_variables::l1_aetsoil
```

Referenced by mo_mhm_eval::mhm_eval(), mo_restart::read_restart_states(), mo_init_states::variables_alloc(), mo_init_states::variables_default_init(), and mo_restart::write_restart_files().

16.22.2.27 l1_baseflow

```
real(dp), dimension(:), allocatable, public mo_global_variables::l1_baseflow
```

Referenced by mo_mhm_eval::mhm_eval(), mo_restart::read_restart_states(), mo_init_states::variables_alloc(), mo_init_states::variables_default_init(), and mo_restart::write_restart_files().

16.22.2.28 l1_et

```
real(dp), dimension(:, :), allocatable, public mo_global_variables::l1_et
```

Referenced by mo_objective_function::objective_et_kge_catchment_avg(), mo_objective_function::objective_kge↩
_q_et(), mo_objective_function::objective_kge_q_rmse_et(), and mo_read_optional_data::read_evapotranspiration().

16.22.2.29 l1_et_mask

```
logical, dimension(:, :), allocatable, public mo_global_variables::l1_et_mask
```

Referenced by mo_objective_function::objective_et_kge_catchment_avg(), mo_objective_function::objective_kge↩
_q_et(), mo_objective_function::objective_kge_q_rmse_et(), and mo_read_optional_data::read_evapotranspiration().

16.22.2.30 l1_fastrunoff

```
real(dp), dimension(:), allocatable, public mo_global_variables::l1_fastrunoff
```

Referenced by mo_mhm_eval::mhm_eval(), mo_restart::read_restart_states(), mo_init_states::variables_alloc(), mo_init_states::variables_default_init(), and mo_restart::write_restart_files().

16.22.2.31 l1_infilsoil

```
real(dp), dimension(:, :), allocatable, public mo_global_variables::l1_infilsoil
```

Referenced by mo_mhm_eval::mhm_eval(), mo_restart::read_restart_states(), mo_init_states::variables_alloc(), mo_init_states::variables_default_init(), and mo_restart::write_restart_files().

16.22.2.32 l1_inter

```
real(dp), dimension(:), allocatable, public mo_global_variables::l1_inter
```

Referenced by mo_mhm_eval::mhm_eval(), mo_restart::read_restart_states(), mo_init_states::variables_alloc(), mo_init_states::variables_default_init(), and mo_restart::write_restart_files().

16.22.2.33 l1_melt

```
real(dp), dimension(:), allocatable, public mo_global_variables::l1_melt
```

Referenced by mo_mhm_eval::mhm_eval(), mo_restart::read_restart_states(), mo_init_states::variables_alloc(), mo_init_states::variables_default_init(), and mo_restart::write_restart_files().

16.22.2.34 l1_netrad

```
real(dp), dimension(:, :), allocatable, public mo_global_variables::l1_netrad
```

Referenced by mo_mhm_eval::mhm_eval(), and mo_meteo_forcings::prepare_meteo_forcings_data().

16.22.2.35 l1_neutrons

```
real(dp), dimension(:), allocatable, public mo_global_variables::l1_neutrons
```

Referenced by mo_mhm_eval::mhm_eval(), mo_init_states::variables_alloc(), and mo_init_states::variables_↩
default_init().

16.22.2.36 l1_neutronsdata

```
real(dp), dimension(:, :), allocatable, public mo_global_variables::l1_neutronsdata
```

Referenced by mo_objective_function::objective_neutrons_kge_catchment_avg(), and mo_read_optional_data↩ ::read_neutrons().

16.22.2.37 l1_neutronsdata_mask

```
logical, dimension(:, :), allocatable, public mo_global_variables::l1_neutronsdata_mask
```

Referenced by mo_objective_function::objective_neutrons_kge_catchment_avg(), and mo_read_optional_data↩ ::read_neutrons().

16.22.2.38 l1_percol

```
real(dp), dimension(:), allocatable, public mo_global_variables::l1_percol
```

Referenced by mo_mhm_eval::mhm_eval(), mo_restart::read_restart_states(), mo_init_states::variables_alloc(), mo_init_states::variables_default_init(), and mo_restart::write_restart_files().

16.22.2.39 l1_pet

```
real(dp), dimension(:, :), allocatable, public mo_global_variables::l1_pet
```

Referenced by mo_mhm_eval::mhm_eval(), and mo_meteo_forcings::prepare_meteo_forcings_data().

16.22.2.40 l1_pet_calc

```
real(dp), dimension(:), allocatable, public mo_global_variables::l1_pet_calc
```

Referenced by mo_mhm_eval::mhm_eval(), mo_init_states::variables_alloc(), and mo_init_states::variables_↩ default_init().

16.22.2.41 l1_pet_weights

```
real(dp), dimension(:, :, :), allocatable, public mo_global_variables::l1_pet_weights
```

Referenced by mo_mhm_eval::mhm_eval(), and mo_meteo_forcings::prepare_meteo_forcings_data().

16.22.2.42 l1_pre

```
real(dp), dimension(:, :), allocatable, public mo_global_variables::l1_pre
```

Referenced by mo_mhm_eval::mhm_eval(), and mo_meteo_forcings::prepare_meteo_forcings_data().

16.22.2.43 l1_pre_weights

```
real(dp), dimension(:, :, :), allocatable, public mo_global_variables::l1_pre_weights
```

Referenced by mo_mhm_eval::mhm_eval(), and mo_meteo_forcings::prepare_meteo_forcings_data().

16.22.2.44 l1_preeffect

```
real(dp), dimension(:), allocatable, public mo_global_variables::l1_preeffect
```

Referenced by mo_mhm_eval::mhm_eval(), mo_restart::read_restart_states(), mo_init_states::variables_alloc(), mo_init_states::variables_default_init(), and mo_restart::write_restart_files().

16.22.2.45 l1_rain

```
real(dp), dimension(:), allocatable, public mo_global_variables::l1_rain
```

Referenced by mo_mhm_eval::mhm_eval(), mo_restart::read_restart_states(), mo_init_states::variables_alloc(), mo_init_states::variables_default_init(), and mo_restart::write_restart_files().

16.22.2.46 l1_runoffseal

```
real(dp), dimension(:), allocatable, public mo_global_variables::l1_runoffseal
```

Referenced by mo_mhm_eval::mhm_eval(), mo_restart::read_restart_states(), mo_init_states::variables_alloc(), mo_init_states::variables_default_init(), and mo_restart::write_restart_files().

16.22.2.47 l1_satstw

```
real(dp), dimension(:), allocatable, public mo_global_variables::l1_satstw
```

Referenced by mo_mhm_eval::mhm_eval(), mo_restart::read_restart_states(), mo_init_states::variables_alloc(), mo_init_states::variables_default_init(), and mo_restart::write_restart_files().

16.22.2.48 l1_sealstw

```
real(dp), dimension(:), allocatable, public mo_global_variables::l1_sealstw
```

Referenced by mo_mhm_eval::mhm_eval(), mo_restart::read_restart_states(), mo_init_states::variables_alloc(), mo_init_states::variables_default_init(), and mo_restart::write_restart_files().

16.22.2.49 l1_slowrunoff

```
real(dp), dimension(:), allocatable, public mo_global_variables::l1_slowrunoff
```

Referenced by mo_mhm_eval::mhm_eval(), mo_restart::read_restart_states(), mo_init_states::variables_alloc(), mo_init_states::variables_default_init(), and mo_restart::write_restart_files().

16.22.2.50 l1_sm

```
real(dp), dimension(:, :), allocatable, public mo_global_variables::l1_sm
```

Referenced by mo_objective_function::objective_kge_q_sm_corr(), mo_objective_function::objective_sm_corr(), mo_objective_function::objective_sm_kge_catchment_avg(), mo_objective_function::objective_sm_pd(), mo_↩ objective_function::objective_sm_sse_standard_score(), and mo_read_optional_data::read_soil_moisture().

16.22.2.51 l1_sm_mask

```
logical, dimension(:, :), allocatable, public mo_global_variables::l1_sm_mask
```

Referenced by mo_objective_function::objective_kge_q_sm_corr(), mo_objective_function::objective_sm_corr(), mo_objective_function::objective_sm_kge_catchment_avg(), mo_objective_function::objective_sm_pd(), mo_↩ objective_function::objective_sm_sse_standard_score(), and mo_read_optional_data::read_soil_moisture().

16.22.2.52 l1_snow

```
real(dp), dimension(:), allocatable, public mo_global_variables::l1_snow
```

Referenced by mo_mhm_eval::mhm_eval(), mo_restart::read_restart_states(), mo_init_states::variables_alloc(), mo_init_states::variables_default_init(), and mo_restart::write_restart_files().

16.22.2.53 l1_snowpack

```
real(dp), dimension(:), allocatable, public mo_global_variables::l1_snowpack
```

Referenced by mo_mhm_eval::mhm_eval(), mo_restart::read_restart_states(), mo_init_states::variables_alloc(), mo_init_states::variables_default_init(), and mo_restart::write_restart_files().

16.22.2.54 l1_soilmoist

```
real(dp), dimension(:, :), allocatable, public mo_global_variables::l1_soilmoist
```

Referenced by mo_mhm_eval::mhm_eval(), mo_restart::read_restart_states(), mo_init_states::variables_alloc(), mo_init_states::variables_default_init(), and mo_restart::write_restart_files().

16.22.2.55 l1_temp

```
real(dp), dimension(:, :), allocatable, public mo_global_variables::l1_temp
```

Referenced by mo_mhm_eval::mhm_eval(), and mo_meteo_forcings::prepare_meteo_forcings_data().

16.22.2.56 l1_temp_weights

```
real(dp), dimension(:, :, :), allocatable, public mo_global_variables::l1_temp_weights
```

Referenced by mo_mhm_eval::mhm_eval(), and mo_meteo_forcings::prepare_meteo_forcings_data().

16.22.2.57 l1_throughfall

```
real(dp), dimension(:), allocatable, public mo_global_variables::l1_throughfall
```

Referenced by mo_mhm_eval::mhm_eval(), mo_restart::read_restart_states(), mo_init_states::variables_alloc(), mo_init_states::variables_default_init(), and mo_restart::write_restart_files().

16.22.2.58 l1_tmax

```
real(dp), dimension(:, :), allocatable, public mo_global_variables::l1_tmax
```

Referenced by mo_mhm_eval::mhm_eval(), and mo_meteo_forcings::prepare_meteo_forcings_data().

16.22.2.59 l1_tmin

```
real(dp), dimension(:, :), allocatable, public mo_global_variables::l1_tmin
```

Referenced by mo_mhm_eval::mhm_eval(), and mo_meteo_forcings::prepare_meteo_forcings_data().

16.22.2.60 l1_total_runoff

```
real(dp), dimension(:), allocatable, public mo_global_variables::l1_total_runoff
```

Referenced by mo_mhm_eval::mhm_eval(), mo_restart::read_restart_states(), mo_init_states::variables_alloc(), mo_init_states::variables_default_init(), and mo_restart::write_restart_files().

16.22.2.61 l1_unsatstw

```
real(dp), dimension(:), allocatable, public mo_global_variables::l1_unsatstw
```

Referenced by mo_mhm_eval::mhm_eval(), mo_restart::read_restart_states(), mo_init_states::variables_alloc(), mo_init_states::variables_default_init(), and mo_restart::write_restart_files().

16.22.2.62 l1_windspeed

```
real(dp), dimension(:, :), allocatable, public mo_global_variables::l1_windspeed
```

Referenced by mo_mhm_eval::mhm_eval(), and mo_meteo_forcings::prepare_meteo_forcings_data().

16.22.2.63 level2

```
type(grid), dimension(:), allocatable, public mo_global_variables::level2
```

Referenced by mo_meteo_forcings::meteo_forcings_wrapper(), mo_meteo_forcings::meteo_weights_wrapper(), and mo_startup::mhm_initialize().

16.22.2.64 neutron_integral_afast

```
real(dp), dimension(:), allocatable, public mo_global_variables::neutron_integral_afast
```

Referenced by mo_startup::constants_init(), and mo_mhm_eval::mhm_eval().

16.22.2.65 nmeasperday_tws

```
integer(i4), public mo_global_variables::nmeasperday_tws
```

Referenced by mo_objective_function::extract_basin_avg_tws(), and mo_read_optional_data::read_basin_avg_↩tws().

16.22.2.66 nsoilhorizons_sm_input

```
integer(i4) mo_global_variables::nsoilhorizons_sm_input
```

Referenced by mo_mhm_eval::mhm_eval(), and mo_mhm_read_config::mhm_read_config().

16.22.2.67 ntimesteps_l1_et

```
integer(i4) mo_global_variables::ntimesteps_l1_et
```

Referenced by mo_mhm_eval::mhm_eval(), and mo_read_optional_data::read_evapotranspiration().

16.22.2.68 ntimesteps_l1_neutrons

```
integer(i4) mo_global_variables::ntimesteps_l1_neutrons
```

Referenced by mo_mhm_eval::mhm_eval(), and mo_read_optional_data::read_neutrons().

16.22.2.69 ntimesteps_l1_sm

```
integer(i4) mo_global_variables::ntimesteps_l1_sm
```

Referenced by mo_mhm_eval::mhm_eval(), and mo_read_optional_data::read_soil_moisture().

16.22.2.70 outputflxstate

```
logical, dimension(noutflxstate) mo_global_variables::outputflxstate
```

Referenced by mo_mhm_eval::mhm_eval(), mo_mhm_read_config::mhm_read_config(), mo_write_fluxes_states↩::newoutputdataset(), and mo_write_fluxes_states::updatedataset().

16.22.2.71 read_meteo_weights

```
logical, public mo_global_variables::read_meteo_weights
```

Referenced by mo_mhm_eval::mhm_eval(), mo_mhm_read_config::mhm_read_config(), and mo_meteo_forcings↩ ::prepare_meteo_forcings_data().

16.22.2.72 routingstates

```
integer(i4), parameter, public mo_global_variables::routingstates = 2
```

16.22.2.73 timestep_et_input

```
integer(i4), public mo_global_variables::timestep_et_input
```

Referenced by mo_mhm_eval::mhm_eval(), mo_mhm_read_config::mhm_read_config(), mo_objective_function↩ ::objective_kge_q_rmse_et(), and mo_read_optional_data::read_evapotranspiration().

16.22.2.74 timestep_model_inputs

```
integer(i4), dimension(:), allocatable, public mo_global_variables::timestep_model_inputs
```

Referenced by mo_meteo_forcings::chunk_size(), mo_meteo_forcings::is_read(), mo_mhm_eval::mhm_eval(), mo_mhm_read_config::mhm_read_config(), and mo_meteo_forcings::prepare_meteo_forcings_data().

16.22.2.75 timestep_model_outputs

```
integer(i4) mo_global_variables::timestep_model_outputs
```

Referenced by mo_write_fluxes_states::fluxesunit(), mo_mhm_eval::mhm_eval(), and mo_mhm_read_config↩ ::mhm_read_config().

16.22.2.76 timestep_neutrons_input

```
integer(i4), public mo_global_variables::timestep_neutrons_input
```

Referenced by mo_mhm_read_config::mhm_read_config(), and mo_read_optional_data::read_neutrons().

16.22.2.77 timestep_sm_input

```
integer(i4), public mo_global_variables::timestep_sm_input
```

Referenced by mo_mhm_eval::mhm_eval(), mo_mhm_read_config::mhm_read_config(), and mo_read_optional_↩ data::read_soil_moisture().

16.23 mo_grid Module Reference

TODO: add description.

Functions/Subroutines

- subroutine, public init_lowres_level (highres, target_resolution, lowres, highres_lowres_remap)

  Level-1 variable initialization.
- subroutine, public set_basin_indices (grids)

  TODO: add description.
- subroutine, public l0_grid_setup (new_grid)

  level 0 variable initialization
- subroutine, public mapcoordinates (level, y, x)

  Generate map coordinates.
- subroutine, public geocoordinates (level, lat, lon)

  Generate geographic coordinates.
- subroutine calculate_grid_properties (nrowsIn, ncolsIn, xllcornerIn, yllcornerIn, cellsizeIn, aimingResolution, nrowsOut, ncolsOut, xllcornerOut, yllcornerOut, cellsizeOut)

  Calculates basic grid properties at a required coarser level using information of a given finer level. Calculates basic grid properties at a required coarser level (e.g., L11) using information of a given finer level (e.g., L0). Basic grid properties such as nrows, ncols, xllcorner, yllcorner cellsize are estimated in this routine.

16.23.1 Detailed Description

TODO: add description.

TODO: add description

Authors

Robert Schweppe

Date

Jun 2018

16.23.2 Function/Subroutine Documentation

16.23.2.1 calculate_grid_properties()

```
subroutine mo_grid::calculate_grid_properties (
             integer(i4), intent(in) nrowsIn,
             integer(i4), intent(in) ncolsIn,
             real(dp), intent(in) xllcornerIn,
             real(dp), intent(in) yllcornerIn,
             real(dp), intent(in) cellsizeIn,
             real(dp), intent(in) aimingResolution,
             integer(i4), intent(out) nrowsOut,
             integer(i4), intent(out) ncolsOut,
             real(dp), intent(out) xllcornerOut,
             real(dp), intent(out) yllcornerOut,
             real(dp), intent(out) cellsizeOut )
```

Calculates basic grid properties at a required coarser level using information of a given finer level. Calculates basic grid properties at a required coarser level (e.g., L11) using information of a given finer level (e.g., L0). Basic grid properties such as nrows, ncols, xllcorner, yllcorner cellsize are estimated in this routine.

TODO: add description

Parameters

in	*integer(i4) :: nrowsIn*	no. of rows at an input level
in	*integer(i4) :: ncolsIn*	no. of cols at an input level
in	*real(dp) :: xllcornerIn*	xllcorner at an input level
in	*real(dp) :: yllcornerIn*	yllcorner at an input level
in	*real(dp) :: cellsizeIn*	cell size at an input level
in	*real(dp) :: aimingResolution*	resolution of an output level
out	*integer(i4) :: nrowsOut*	no. of rows at an output level
out	*integer(i4) :: ncolsOut*	no. of cols at an output level
out	*real(dp) :: xllcornerOut*	xllcorner at an output level
out	*real(dp) :: yllcornerOut*	yllcorner at an output level
out	*real(dp) :: cellsizeOut*	cell size at an output level

Authors

Matthias Zink & Rohini Kumar

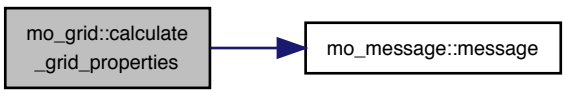
Date

Feb 2013

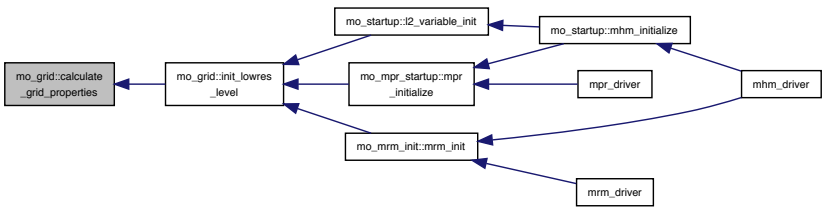
References mo_message::message().

Referenced by init_lowres_level().

Here is the call graph for this function:



Here is the caller graph for this function:



16.23.2.2 geocoordinates()

```
subroutine, public mo_grid::geocoordinates (
            type(grid), intent(in) level,
            real(dp), dimension(:, :), intent(out), allocatable lat,
            real(dp), dimension(:, :), intent(out), allocatable lon )
```

Generate geographic coordinates.

Generate geographic coordinate arrays for given basin and level

Parameters

in	*type(Grid) :: level*	-> grid reference
out	*real(dp), dimension(:, :) :: lat, lon*	
out	*real(dp), dimension(:, :) :: lat, lon*	

Authors

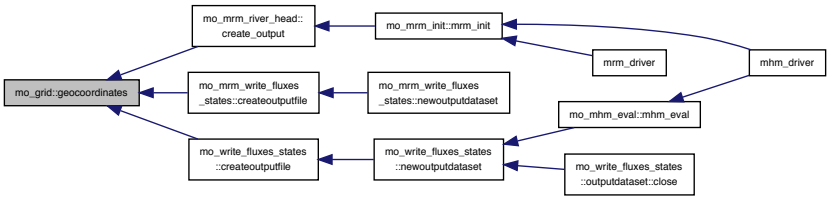
Matthias Zink

Date

Apr 2013

Referenced by mo_mrm_river_head::create_output(), mo_mrm_write_fluxes_states::createoutputfile(), and mo_↩ write_fluxes_states::createoutputfile().

Here is the caller graph for this function:



16.23.2.3 init_lowres_level()

```
subroutine, public mo_grid::init_lowres_level (
            type(grid), intent(in), target highres,
            real(dp), intent(in) target_resolution,
            type(grid), intent(inout), target lowres,
            type(gridremapper), intent(inout), optional highres_lowres_remap )
```

Level-1 variable initialization.

following tasks are performed for L1 datasets

- cell id & numbering

- mask creation
- storage of cell cordinates (row and coloum id)
- sorage of four corner L0 cordinates If a variable is added or removed here, then it also has to be added or removed in the subroutine config_variables_set in module mo_restart and in the subroutine set_config in module mo_set_netcdf_restart

Parameters

in	*type(Grid) :: highres*	
in	*real(dp) :: target_resolution*	
in,out	*type(Grid) :: lowres*	
in,out	*type(GridRemapper), optional :: highres_lowres_remap*	

Authors

Rohini Kumar

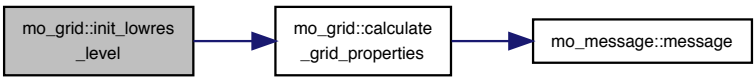
Date

Jan 2013

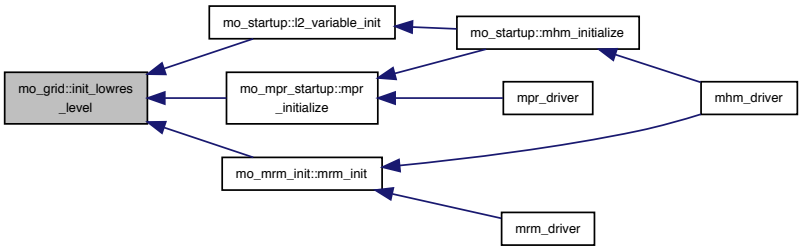
References calculate_grid_properties(), mo_kind::i4, mo_common_constants::nodata_dp, and mo_common_↩ constants::nodata_i4.

Referenced by mo_startup::l2_variable_init(), mo_mpr_startup::mpr_initialize(), and mo_mrm_init::mrm_init().

Here is the call graph for this function:



Here is the caller graph for this function:



16.23.2.4 l0_grid_setup()

```
subroutine, public mo_grid::l0_grid_setup (
             type(grid), intent(inout) new_grid )
```

level 0 variable initialization

following tasks are performed for L0 data sets

- cell id & numbering
- storage of cell cordinates (row and coloum id)
- empirical dist. of terrain slope
- flag to determine the presence of a particular soil id in this configuration of the model run If a variable is added or removed here, then it also has to be added or removed in the subroutine config_variables_set in module mo_restart and in the subroutine set_config in module mo_set_netcdf_restart

Parameters

in,out	*type(Grid) :: new_grid*	

Authors

Rohini Kumar

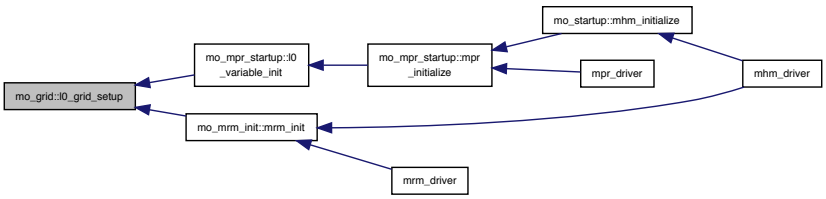
Date

Jan 2013

References mo_common_variables::iflag_cordinate_sys, mo_constants::radiusearth_dp, and mo_constants::twopi_dp.

Referenced by mo_mpr_startup::l0_variable_init(), and mo_mrm_init::mrm_init().

Here is the caller graph for this function:



16.23.2.5 mapcoordinates()

```
subroutine, public mo_grid::mapcoordinates (
             type(grid), intent(in) level,
             real(dp), dimension(:), intent(out), allocatable y,
             real(dp), dimension(:), intent(out), allocatable x )
```

Generate map coordinates.

Generate map coordinate arrays for given basin and level

Parameters

in	*type(Grid) :: level*	-> grid reference
out	*real(dp), dimension(:) :: x, y*	
out	*real(dp), dimension(:) :: x, y*	

Authors

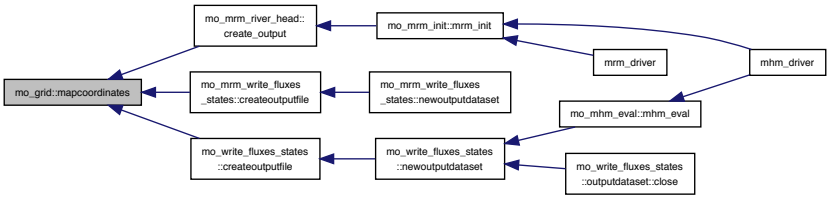
Matthias Zink

Date

Apr 2013

Referenced by mo_mrm_river_head::create_output(), mo_mrm_write_fluxes_states::createoutputfile(), and mo_write_fluxes_states::createoutputfile().

Here is the caller graph for this function:



16.23.2.6 set_basin_indices()

```
subroutine, public mo_grid::set_basin_indices (
             type(grid), dimension(:), intent(inout) grids )
```

TODO: add description.

TODO: add description

Parameters

in,out	*type(Grid), dimension(:) :: grids*	

Authors

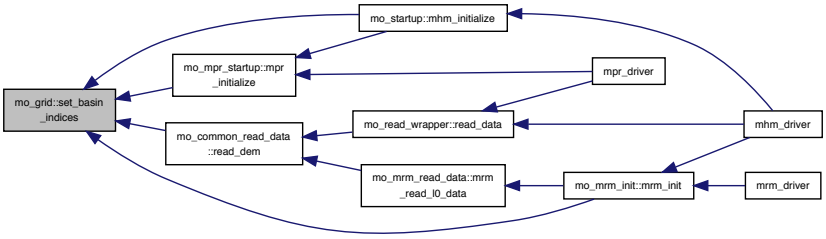
Robert Schweppe

Date

Jun 2018

Referenced by mo_startup::mhm_initialize(), mo_mpr_startup::mpr_initialize(), mo_mrm_init::mrm_init(), and mo_common_read_data::read_dem().

Here is the caller graph for this function:



16.24 mo_init_states Module Reference

Initialization of all state variables of mHM.

Functions/Subroutines

- subroutine, public variables_alloc (ncells1)

  Allocation of space for mHM related L1 and L11 variables.
- subroutine, public variables_default_init

  Default initalization mHM related L1 variables.

16.24.1 Detailed Description

Initialization of all state variables of mHM.

This module initializes all state variables required to run mHM. Two options are provided:

- (1) default values
- (2) from nc file

Authors

Luis Samaniego & Rohini Kumar

Date

Dec 2012

16.24.2 Function/Subroutine Documentation

16.24.2.1 variables_alloc()

```
subroutine, public mo_init_states::variables_alloc (
            integer(i4), intent(in) ncells1 )
```

Allocation of space for mHM related L1 and L11 variables.

Allocation of space for mHM related L1 and L11 variables (e.g., states, fluxes, and parameters) for a given basin. Variables allocated here is defined in them mo_global_variables.f90 file. After allocating any variable in this routine, initalize them in the following variables_default_init subroutine:

Parameters

in	*integer(i4) :: ncells1*	

Authors

Rohini Kumar

Date

Jan 2013

References mo_mpr_constants::c1_initstatesm, mo_mpr_global_variables::horizondepth_mhm, mo_global_variables::l1_aetcanopy, mo_global_variables::l1_aetsealed, mo_global_variables::l1_aetsoil, mo_global_variables::l1_baseflow, mo_global_variables::l1_fastrunoff, mo_global_variables::l1_infilsoil, mo_global_variables::l1_inter, mo_global_variables::l1_melt, mo_global_variables::l1_neutrons, mo_global_variables::l1_percol, mo_global_variables::l1_pet_calc, mo_global_variables::l1_preeffect, mo_global_variables::l1_rain, mo_global_variables::l1_runoffseal, mo_global_variables::l1_satstw, mo_global_variables::l1_sealstw, mo_global_variables::l1_slowrunoff, mo_global_variables::l1_snow, mo_global_variables::l1_snowpack, mo_global_variables::l1_soilmoist, mo_global_variables::l1_throughfall, mo_global_variables::l1_total_runoff, mo_global_variables::l1_unsatstw, mo_mpr_global_variables::nsoilhorizons_mhm, mo_common_constants::p1_initstatefluxes, mo_mpr_constants::p2_initstatefluxes, mo_mpr_constants::p3_initstatefluxes, mo_mpr_constants::p4_initstatefluxes, and mo_mpr_constants::p5_initstatefluxes.

Referenced by mo_startup::mhm_initialize().

Here is the caller graph for this function:

16.24.2.2 variables_default_init()

```
subroutine, public mo_init_states::variables_default_init ( )
```

Default initalization mHM related L1 variables.

Default initalization of mHM related L1 variables (e.g., states, fluxes, and parameters) as per given constant values given in mo_mhm_constants. Variables initalized here is defined in the mo_global_variables.f90 file. Only Variables that are defined in the variables_alloc subroutine are intialized here. If a variable is added or removed here, then it also has to be added or removed in the subroutine state_variables_set in the module mo_restart and in the subroutine set_state in the module mo_set_netcdf_restart.

Authors

R. Kumar & J. Mai

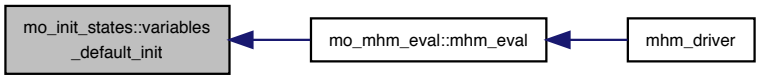
Date

Sep 2013

References mo_mpr_constants::c1_initstatesm, mo_mpr_global_variables::horizondepth_mhm, mo_mpr_global_variables::l1_aeroresist, mo_global_variables::l1_aetcanopy, mo_global_variables::l1_aetsealed, mo_global_variables::l1_aetsoil, mo_mpr_global_variables::l1_alpha, mo_global_variables::l1_baseflow, mo_mpr_global_variables::l1_degday, mo_mpr_global_variables::l1_degdayinc, mo_mpr_global_variables::l1_degdaymax, mo_mpr_global_variables::l1_degdaynopre, mo_mpr_global_variables::l1_fasp, mo_global_variables::l1_fastrunoff, mo_mpr_global_variables::l1_froots, mo_mpr_global_variables::l1_fsealed, mo_mpr_global_variables::l1_harsamcoeff, mo_global_variables::l1_infilsoil, mo_global_variables::l1_inter, mo_mpr_global_variables::l1_jarvis_thresh_c1, mo_mpr_global_variables::l1_karstloss, mo_mpr_global_variables::l1_kbaseflow, mo_mpr_global_variables::l1_kfastflow, mo_mpr_global_variables::l1_kperco, mo_mpr_global_variables::l1_kslowflow, mo_mpr_global_variables::l1_maxinter, mo_global_variables::l1_melt, mo_global_variables::l1_neutrons, mo_global_variables::l1_percol, mo_global_variables::l1_pet_calc, mo_mpr_global_variables::l1_petlaicorfactor, mo_global_variables::l1_preeffect, mo_mpr_global_variables::l1_prietayalpha, mo_global_variables::l1_rain, mo_global_variables::l1_runoffseal, mo_global_variables::l1_satstw, mo_mpr_global_variables::l1_sealedthresh, mo_global_variables::l1_sealstw, mo_global_variables::l1_slowrunoff, mo_global_variables::l1_snow, mo_global_variables::l1_snowpack, mo_global_variables::l1_soilmoist, mo_mpr_global_variables::l1_soilmoistexp, mo_mpr_global_variables::l1_soilmoistfc, mo_mpr_global_variables::l1_soilmoistsat, mo_mpr_global_variables::l1_surfresist, mo_mpr_global_variables::l1_tempthresh, mo_global_variables::l1_throughfall, mo_global_variables::l1_total_runoff, mo_global_variables::l1_unsatstw, mo_mpr_global_variables::l1_unsatthresh, mo_mpr_global_variables::l1_wiltingpoint, mo_mpr_global_variables::nsoilhorizons_mhm, mo_common_constants::p1_initstatefluxes, mo_mpr_constants::p2_initstatefluxes, mo_mpr_constants::p3_initstatefluxes, mo_mpr_constants::p4_initstatefluxes, and mo_mpr_constants::p5_initstatefluxes.

Referenced by mo_mhm_eval::mhm_eval().

Here is the caller graph for this function:



16.25 mo_julian Module Reference

Julian date conversion routines.

Data Types

- interface setcalendar

Functions/Subroutines

- subroutine setcalendarstring (selector)

  Set module private variable calendar.
- subroutine setcalendarinteger (selector)

  Set module private variable calendar.
- pure integer(i4) function selectcalendar (selector)

  Select a calendar.

- elemental subroutine, public caldat (julian, dd, mm, yy, calendar)

  Day, month and year from Julian day in the current or given calendar.
- elemental subroutine, public dec2date (julian, dd, mm, yy, hh, nn, ss, calendar)

  Day, month, year, hour, minute, and second from fractional Julian day in the current or given calendar.
- elemental real(dp) function, public date2dec (dd, mm, yy, hh, nn, ss, calendar)

  Fractional Julian day from day, month, year, hour, minute, second in the current calendar.
- elemental integer(i4) function, public julday (dd, mm, yy, calendar)

  Julian day from day, month and year in the current or given calendar.
- elemental subroutine, public caldatjulian (julian, dd, mm, yy)

  Day, month and year from Julian day.
- elemental real(dp) function date2decjulian (dd, mm, yy, hh, nn, ss)

  Fractional Julian day from day, month, year, hour, minute, second.
- elemental subroutine dec2datejulian (julian, dd, mm, yy, hh, nn, ss)

  Day, month, year, hour, minute, and second from fractional Julian day.
- elemental integer(i4) function juldayjulian (dd, mm, yy)

  Julian day from day, month and year.
- elemental integer(i4) function, public ndays (dd, mm, yy)

  IMSL Julian day from day, month and year.
- elemental subroutine, public ndyin (julian, dd, mm, yy)

  Day, month and year from IMSL Julian day.
- elemental subroutine caldat360 (julian, dd, mm, yy)

  Day, month and year from Julian day in a 360 day calendar.
- elemental integer(i4) function julday360 (dd, mm, yy)

  Julian day from day, month and year in a 360_day calendar.
- elemental subroutine dec2date360 (julian, dd, mm, yy, hh, nn, ss)

  Day, month, year, hour, minute, and second from fractional Julian day in a 360_day calendar.
- elemental real(dp) function date2dec360 (dd, mm, yy, hh, nn, ss)

  Fractional Julian day from day, month, year, hour, minute, second in 360 day calendar.
- elemental subroutine caldat365 (julian, dd, mm, yy)

  Day, month and year from Julian day in a 365 day calendar.
- elemental integer(i4) function julday365 (dd, mm, yy)

  Julian day from day, month and year in a 365_day calendar.
- elemental subroutine dec2date365 (julian, dd, mm, yy, hh, nn, ss)

  Day, month, year, hour, minute, and second from fractional Julian day in a 365_day calendar.
- elemental real(dp) function date2dec365 (dd, mm, yy, hh, nn, ss)

  Fractional Julian day from day, month, year, hour, minute, second in 365 day calendar.

Variables

- integer(i4), save, private calendar = 1

16.25.1 Detailed Description

Julian date conversion routines.

Julian date to and from day, month, year, and also from day, month, year, hour, minute, and second. Also convience routines for Julian dates of IMSL are provided.

Note

Julian day definition starts at noon of the 1st January 4713 BC.
Here, the astronomical definition is used, i.e. the year 1 BC (historic) is counted as 0 (astronomic), 2 BC is -1, etc.
This means that Julian day definition starts as 01.01.-4712 in astronomical units.

julday and caldat start at midnight of the 1st January 4713 BC. So date2dec and julday as well as dec2date and caldat are shifted by half a day.
Use date2dec with dec2date together for fractional Julian dates and use julday with caldat together for integer Julian days.

Author

Matthias Cuntz

Date

Dec 2011

16.25.2 Function/Subroutine Documentation

16.25.2.1 caldat()

```
elemental subroutine, public mo_julian::caldat (
             integer(i4), intent(in) julian,
             integer(i4), intent(out) dd,
             integer(i4), intent(out) mm,
             integer(i4), intent(out) yy,
             integer(i4), intent(in), optional calendar )
```

Day, month and year from Julian day in the current or given calendar.

Wrapper around the calendar specific caldat procedures. Inverse of the function julday. Here julian is input as a Julian Day Number, and the routine outputs d0d, mm, and yy as the day, month, and year on which the specified Julian Day started at noon. The zeroth Julian Day depends on the called procedure. See their documentation for details.

Parameters

in	*integer(i4) :: julday*	Julian day
out	*integer(i4) :: dd*	Day in month of Julian day
out	*integer(i4) :: mm*	Month in year of Julian day
out	*integer(i4) :: yy*	Year of Julian day
in	*integer(i4) :: calendar*	The calendar to use, the global calendar will be used by default

Author

Written, David Schaefer

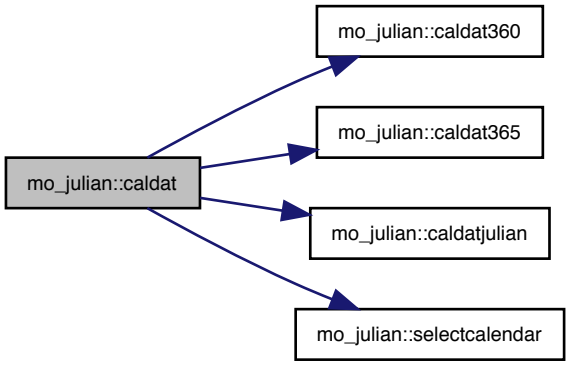
Date

Jan 2015

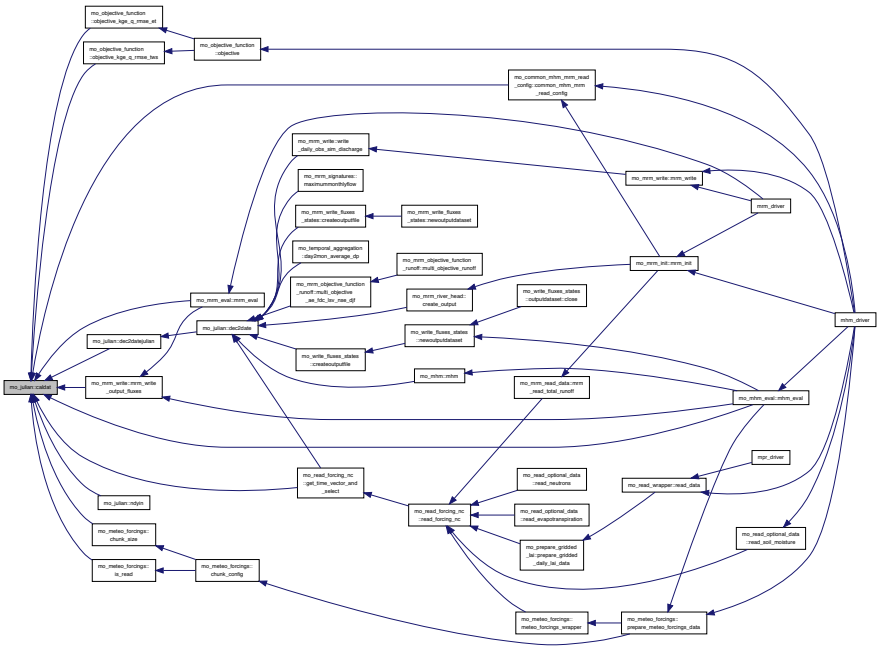
References caldat360(), caldat365(), caldatjulian(), and selectcalendar().

Referenced by mo_meteo_forcings::chunk_size(), mo_common_mhm_mrm_read_config::common_mhm_mrm_read_config(), dec2datejulian(), mo_read_forcing_nc::get_time_vector_and_select(), mo_meteo_forcings::is_read(), mo_mhm_eval::mhm_eval(), mo_mrm_eval::mrm_eval(), mo_mrm_write::mrm_write_output_fluxes(), ndyin(), mo_objective_function::objective_kge_q_rmse_et(), and mo_objective_function::objective_kge_q_rmse_tws().

Here is the call graph for this function:



Here is the caller graph for this function:



16.25.2.2 caldat360()

```
elemental subroutine mo_julian::caldat360 (
            integer(i4), intent(in) julian,
            integer(i4), intent(out) dd,
            integer(i4), intent(out) mm,
            integer(i4), intent(out) yy )  [private]
```

Day, month and year from Julian day in a 360 day calendar.

Inverse of the function julday360. Here julian is input as a Julian Day Number, and the routine outputs dd, mm, and yy as the day, month, and year on which the specified Julian Day started at noon. The zeroth Julian Day here is 01.01.0000

Parameters

in	*integer(i4) :: julday*	Julian day
out	*integer(i4) :: dd*	Day in month of Julian day
out	*integer(i4) :: mm*	Month in year of Julian day
out	*integer(i4) :: yy*	Year of Julian day

Author

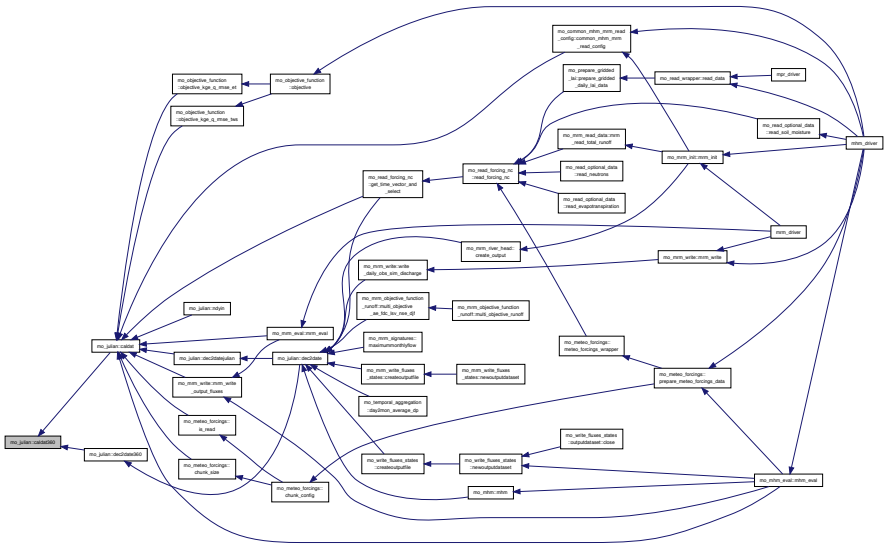
Written, David Schaefer

Date

Oct 2015

Referenced by caldat(), and dec2date360().

Here is the caller graph for this function:



16.25.2.3 caldat365()

```
elemental subroutine mo_julian::caldat365 (
            integer(i4), intent(in) julian,
            integer(i4), intent(out) dd,
            integer(i4), intent(out) mm,
            integer(i4), intent(out) yy )  [private]
```

Day, month and year from Julian day in a 365 day calendar.

Inverse of the function julday365. Here julian is input as a Julian Day Number, and the routine outputs dd, mm, and yy as the day, month, and year on which the specified Julian Day started at noon. The zeroth Julian Day here is 01.01.0000

Parameters

in	*integer(i4) :: julday*	Julian day
out	*integer(i4) :: dd*	Day in month of Julian day
out	*integer(i4) :: mm*	Month in year of Julian day
out	*integer(i4) :: yy*	Year of Julian day

Author

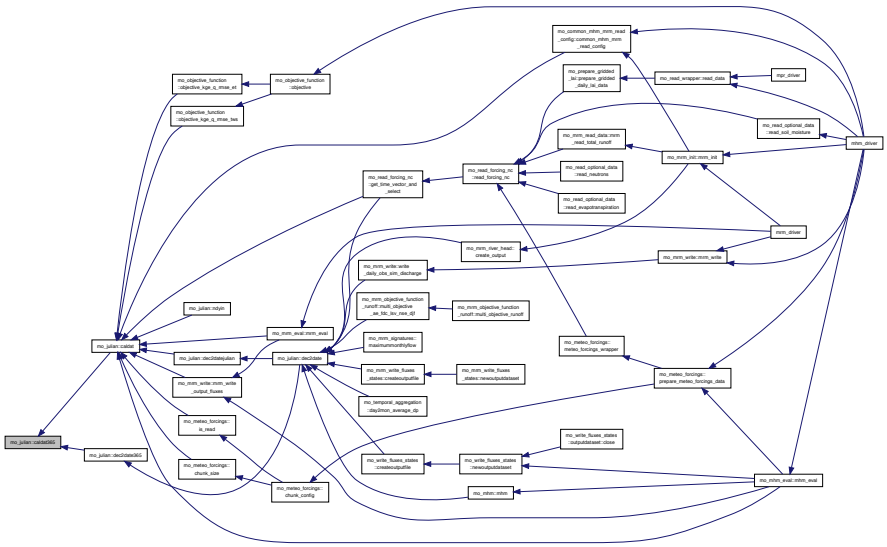
Written, David Schaefer

Date

Dec 2015

Referenced by caldat(), and dec2date365().

Here is the caller graph for this function:



16.25.2.4 caldatjulian()

```
elemental subroutine, public mo_julian::caldatjulian (
            integer(i4), intent(in) julian,
            integer(i4), intent(out) dd,
            integer(i4), intent(out) mm,
            integer(i4), intent(out) yy )
```

Day, month and year from Julian day.

Inverse of the function juldayJulian. Here julian is input as a Julian Day Number, and the routine outputs id, mm, and yy as the day, month, and year on which the specified Julian Day started at noon. The zeroth Julian Day is 01.01.-4712, i.e. the 1st January 4713 BC. Julian day definition starts at 1st January 4713 BC.
Here, the astronomical definition is used, i.e. the year 1 BC (historic) is counted as 0 (astronomic), 2 BC is -1, etc.
This means that Julian day definition starts as 01.01.-4712 in astronomical units.

Parameters

in	*integer(i4) :: Julday*	Julian day
out	*integer(i4) :: dd*	Day in month of Julian day
out	*integer(i4) :: mm*	Month in year of Julian day
out	*integer(i4) :: yy*	Year of Julian day

Note

Julian day definition starts at noon of the 1st January 4713 BC.
Here, the astronomical definition is used, i.e. the year 1 BC (historic) is counted as 0 (astronomic), 2 BC is -1, etc.
This means that Julian day definition starts as 01.01.-4712 in astronomical units.

julday and caldat start at midnight of the 1st January 4713 BC. So date2decJulian and juldayJulian as well as dec2dateJulian and caldatJulian are shifted by half a day.
Use date2decJulian with dec2dateJulian together for fractional Julian dates and use juldayJulian with caldat↩ Julian together for integer Julian days.

Author

Written, Matthias Cuntz - modified julday from Numerical Recipes

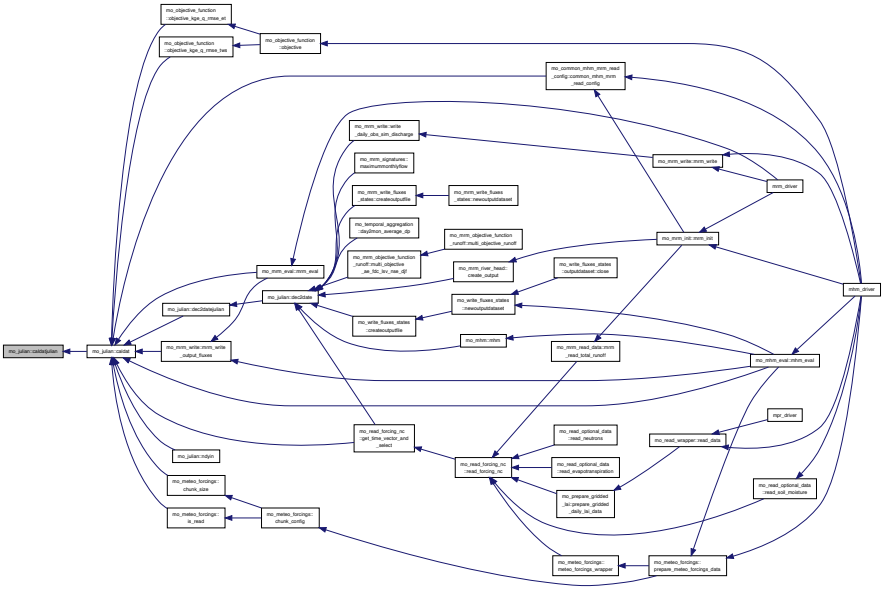
Date

Dec 2011 Modified Matthias Cuntz, May 2014 - changed to new algorithm with astronomical units removed numerical recipes David Schaefer, Jan 2016 - renamed procodure

References mo_kind::i4, and mo_kind::i8.

Referenced by caldat().

Here is the caller graph for this function:



16.25.2.5 date2dec()

```
elemental real(dp) function, public mo_julian::date2dec (
            integer(i4), intent(in), optional dd,
            integer(i4), intent(in), optional mm,
            integer(i4), intent(in), optional yy,
            integer(i4), intent(in), optional hh,
            integer(i4), intent(in), optional nn,
            integer(i4), intent(in), optional ss,
            integer(i4), intent(in), optional calendar )
```

Fractional Julian day from day, month, year, hour, minute, second in the current calendar.

Wrapper around the calendar specific date2dec procedures. In this routine date2dec returns the fractional Julian Day that begins at noon of the calendar date specified by month mm, day dd, and year yy, all integer variables. The zeroth Julian Day depends on the called procedure. See their documentation for details.

Parameters

in	*integer(i4), optional :: dd*	Day in month of Julian day (default: 1)
in	*integer(i4), optional :: mm*	Month in year of Julian day (default: 1)
in	*integer(i4), optional :: yy*	Year of Julian day (default: 1)
in	*integer(i4), optional :: hh*	Hours of Julian day (default: 0)
in	*integer(i4), optional :: nn*	Minutes of hour of Julian day (default: 0)

Parameters

in	*integer(i4), optional :: ss*	Secondes of minute of hour of Julian day (default: 0)
in	*integer(i4), optional :: calendar*	The calendar to use, the global calendar will be used by default

Returns

real(dp) :: date2dec ! Fractional Julian day

Author

Written, David Schaefer

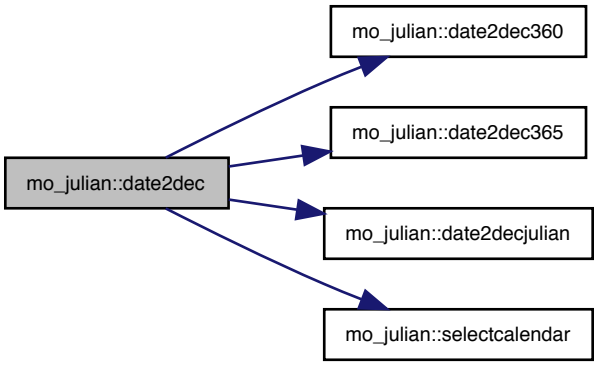
Date

Jan 2015

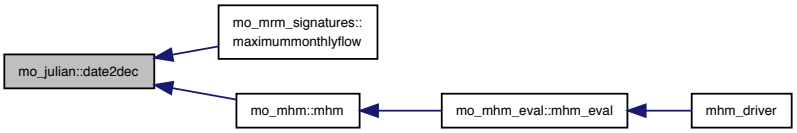
References date2dec360(), date2dec365(), date2decjulian(), and selectcalendar().

Referenced by mo_mrm_signatures::maximummonthlyflow(), and mo_mhm::mhm().

Here is the call graph for this function:



Here is the caller graph for this function:



16.25.2.6 date2dec360()

```
elemental real(dp) function mo_julian::date2dec360 (
            integer(i4), intent(in), optional dd,
            integer(i4), intent(in), optional mm,
            integer(i4), intent(in), optional yy,
            integer(i4), intent(in), optional hh,
            integer(i4), intent(in), optional nn,
            integer(i4), intent(in), optional ss )  [private]
```

Fractional Julian day from day, month, year, hour, minute, second in 360 day calendar.

In this routine date2dec360 returns the fractional Julian Day that begins at noon of the calendar date specified by month mm, day dd, and year yy, all integer variables. The zeroth Julian Day is 01.01.0000 at noon.

Parameters

in	*integer(i4), optional :: dd*	Day in month of Julian day (default: 1)
in	*integer(i4), optional :: mm*	Month in year of Julian day (default: 1)
in	*integer(i4), optional :: yy*	Year of Julian day (default: 1)
in	*integer(i4), optional :: hh*	Hours of Julian day (default: 0)
in	*integer(i4), optional :: nn*	Minutes of hour of Julian day (default: 0)
in	*integer(i4), optional :: ss*	Secondes of minute of hour of Julian day (default: 0)

Returns

real(dp) :: date2dec360 ! Fractional Julian day

Author

Written, David Schaefer

Date

Oct 2015

Referenced by date2dec().

Here is the caller graph for this function:

16.25.2.7 date2dec365()

```
elemental real(dp) function mo_julian::date2dec365 (
            integer(i4), intent(in), optional dd,
            integer(i4), intent(in), optional mm,
```

```
            integer(i4), intent(in), optional yy,
            integer(i4), intent(in), optional hh,
            integer(i4), intent(in), optional nn,
            integer(i4), intent(in), optional ss )  [private]
```

Fractional Julian day from day, month, year, hour, minute, second in 365 day calendar.

In this routine date2dec365 returns the fractional Julian Day that begins at noon of the calendar date specified by month mm, day dd, and year yy, all integer variables. The zeroth Julian Day is 01.01.0000 at noon.

Parameters

in	*integer(i4), optional :: dd*	Day in month of Julian day (default: 1)
in	*integer(i4), optional :: mm*	Month in year of Julian day (default: 1)
in	*integer(i4), optional :: yy*	Year of Julian day (default: 1)
in	*integer(i4), optional :: hh*	Hours of Julian day (default: 0)
in	*integer(i4), optional :: nn*	Minutes of hour of Julian day (default: 0)
in	*integer(i4), optional :: ss*	Secondes of minute of hour of Julian day (default: 0)

Returns

real(dp) :: date2dec365 ! Fractional Julian day

Author

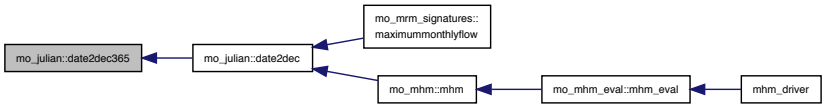
Written, David Schaefer

Date

Dec 2015

Referenced by date2dec().

Here is the caller graph for this function:



16.25.2.8 date2decjulian()

```
elemental real(dp) function mo_julian::date2decjulian (
            integer(i4), intent(in), optional dd,
            integer(i4), intent(in), optional mm,
            integer(i4), intent(in), optional yy,
            integer(i4), intent(in), optional hh,
            integer(i4), intent(in), optional nn,
            integer(i4), intent(in), optional ss )  [private]
```

Fractional Julian day from day, month, year, hour, minute, second.

In this routine date2decJulian returns the fractional Julian Day that begins at noon of the calendar date specified by month mm, day dd, and year yy, all integer variables. The zeroth Julian Day is 01.01.-4712 at noon, i.e. the 1st January 4713 BC 12:00:00 h. Julian day definition starts at noon of the 1st January 4713 BC.
Here, the astronomical definition is used, i.e. the year 1 BC (historic) is counted as 0 (astronomic), 2 BC is -1, etc.
This means that Julian day definition starts as 01.01.-4712 in astronomical units.

Parameters

in	*integer(i4), optional :: dd*	Day in month of Julian day (default: 1)
in	*integer(i4), optional :: mm*	Month in year of Julian day (default: 1)
in	*integer(i4), optional :: yy*	Year of Julian day (default: 1)
in	*integer(i4), optional :: hh*	Hours of Julian day (default: 0)
in	*integer(i4), optional :: nn*	Minutes of hour of Julian day (default: 0)
in	*integer(i4), optional :: ss*	Secondes of minute of hour of Julian day (default: 0)

Returns

real(dp) :: date2dec — Fractional Julian day

Note

Julian day definition starts at noon of the 1st January 4713 BC.
Here, the astronomical definition is used, i.e. the year 1 BC (historic) is counted as 0 (astronomic), 2 BC is -1, etc.
This means that Julian day definition starts as 01.01.-4712 in astronomical units.

juldayJulian and caldatJulian start at midnight of the 1st January 4713 BC. So date2decJulian and juldayJulian as well as dec2dateJulian and caldatJulian are shifted by half a day.
Use date2decJulian with dec2dateJulian together for fractional Julian dates and use juldayJulian with caldat↩ Julian together for integer Julian days.

Author

Written, Matthias Cuntz

Date

Jan 2013 Modified Matthias Cuntz, May 2014 - changed to new algorithm with astronomical units removed numerical recipes David Schaefer, Jan 2016 - renamed procodure

References mo_kind::i8.

Referenced by date2dec().

Here is the caller graph for this function:

16.25.2.9 dec2date()

```
elemental subroutine, public mo_julian::dec2date (
            real(dp), intent(in) julian,
            integer(i4), intent(out), optional dd,
            integer(i4), intent(out), optional mm,
            integer(i4), intent(out), optional yy,
            integer(i4), intent(out), optional hh,
            integer(i4), intent(out), optional nn,
            integer(i4), intent(out), optional ss,
            integer(i4), intent(in), optional calendar )
```

Day, month, year, hour, minute, and second from fractional Julian day in the current or given calendar.

Wrapper around the calendar specific dec2date procedures. Inverse of the function date2dec. Here dec2date is input as a fractional Julian Day. The routine outputs dd, mm, yy, hh, nn, ss as the day, month, year, hour, minute, and second on which the specified Julian Day started at noon. The zeroth Julian Day depends on the called procedure. See their documentation for details.

Parameters

in	*real(dp) :: fJulian*	fractional Julian day
in	*integer(i4) :: calendar*	The calendar to use, the global calendar will be used by default
out	*integer(i4), optional :: dd*	Day in month of Julian day
out	*integer(i4), optional :: mm*	Month in year of Julian day
out	*integer(i4), optional :: yy*	Year of Julian day
out	*integer(i4), optional :: hh*	Hour of Julian day
out	*integer(i4), optional :: nn*	Minute in hour of Julian day
out	*integer(i4), optional :: ss*	Second in minute of hour of Julian day

Author

Written, David Schaefer

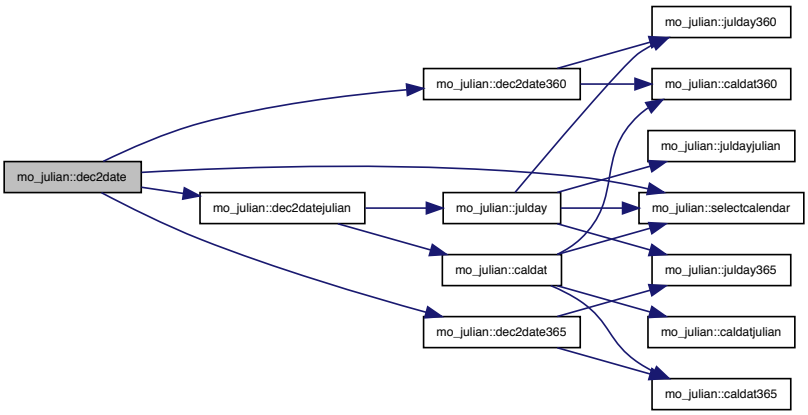
Date

Jan 2015

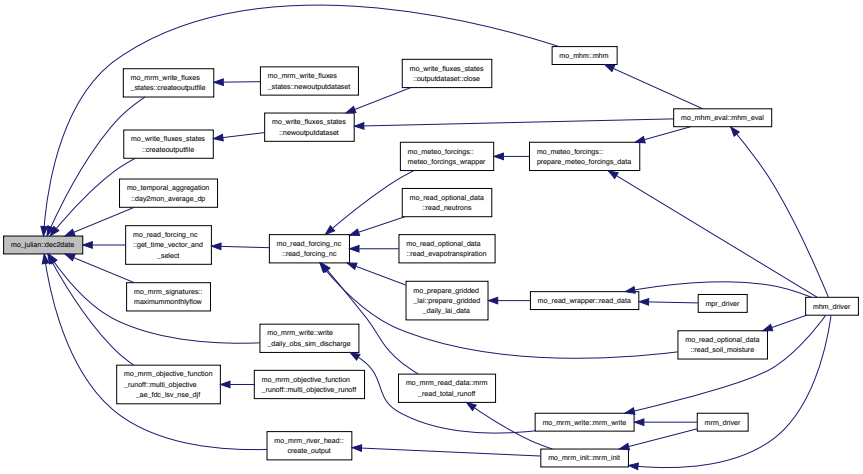
References dec2date360(), dec2date365(), dec2datejulian(), and selectcalendar().

Referenced by mo_mrm_river_head::create_output(), mo_mrm_write_fluxes_states::createoutputfile(), mo_write_fluxes_states::createoutputfile(), mo_temporal_aggregation::day2mon_average_dp(), mo_read_forcing_nc::get_time_vector_and_select(), mo_mrm_signatures::maximummonthlyflow(), mo_mhm::mhm(), mo_mrm_objective_function_runoff::multi_objective_ae_fdc_lsv_nse_djf(), and mo_mrm_write::write_daily_obs_sim_discharge().

Here is the call graph for this function:



Here is the caller graph for this function:



16.25.2.10 dec2date360()

```
elemental subroutine mo_julian::dec2date360 (
            real(dp), intent(in) julian,
            integer(i4), intent(out), optional dd,
            integer(i4), intent(out), optional mm,
            integer(i4), intent(out), optional yy,
            integer(i4), intent(out), optional hh,
            integer(i4), intent(out), optional nn,
            integer(i4), intent(out), optional ss )  [private]
```

Day, month, year, hour, minute, and second from fractional Julian day in a 360_day calendar.

Inverse of the function date2dec360. Here dec2date360 is input as a fractional Julian Day. The routine outputs dd,

mm, yy, hh, nn, ss as the day, month, year, hour, minute, and second on which the specified Julian Day started at noon. The zeroth Julian Day is 01.01.0000 at noon.

Parameters

in	*real(dp) :: fJulian*	fractional Julian day
out	*integer(i4), optional :: dd*	Day in month of Julian day
out	*integer(i4), optional :: mm*	Month in year of Julian day
out	*integer(i4), optional :: yy*	Year of Julian day
out	*integer(i4), optional :: hh*	Hour of Julian day
out	*integer(i4), optional :: nn*	Minute in hour of Julian day
out	*integer(i4), optional :: ss*	Second in minute of hour of Julian day

Author

Written, David Schaefer

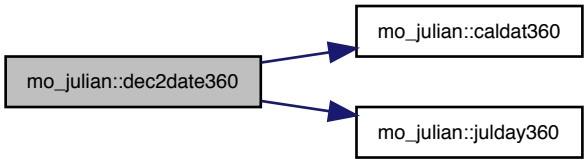
Date

Oct 2015

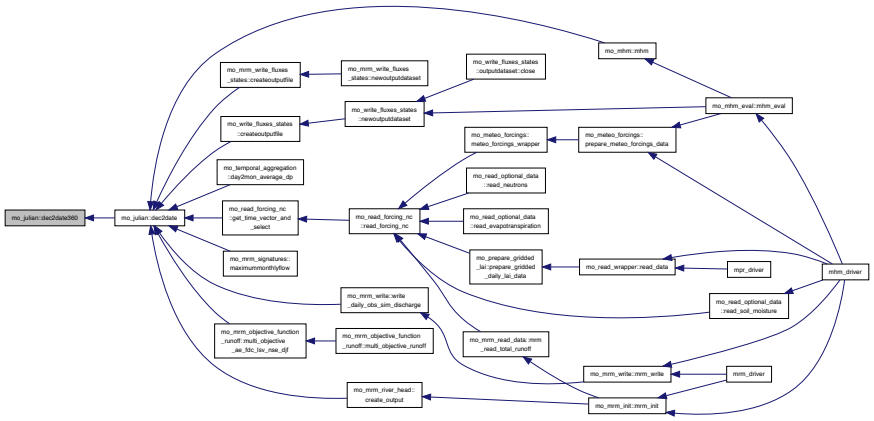
References caldat360(), mo_kind::i4, and julday360().

Referenced by dec2date().

Here is the call graph for this function:



Here is the caller graph for this function:



16.25.2.11 dec2date365()

```
elemental subroutine mo_julian::dec2date365 (
            real(dp), intent(in) julian,
            integer(i4), intent(out), optional dd,
            integer(i4), intent(out), optional mm,
            integer(i4), intent(out), optional yy,
            integer(i4), intent(out), optional hh,
            integer(i4), intent(out), optional nn,
            integer(i4), intent(out), optional ss )  [private]
```

Day, month, year, hour, minute, and second from fractional Julian day in a 365_day calendar.

Inverse of the function date2dec. Here dec2date365 is input as a fractional Julian Day. The routine outputs dd, mm, yy, hh, nn, ss as the day, month, year, hour, minute, and second on which the specified Julian Day started at noon. The zeroth Julian Day is 01.01.0000 at noon.

Parameters

in	*real(dp) :: fJulian*	fractional Julian day
out	*integer(i4), optional :: dd*	Day in month of Julian day
out	*integer(i4), optional :: mm*	Month in year of Julian day
out	*integer(i4), optional :: yy*	Year of Julian day
out	*integer(i4), optional :: hh*	Hour of Julian day
out	*integer(i4), optional :: nn*	Minute in hour of Julian day
out	*integer(i4), optional :: ss*	Second in minute of hour of Julian day

Author

Written, David Schaefer

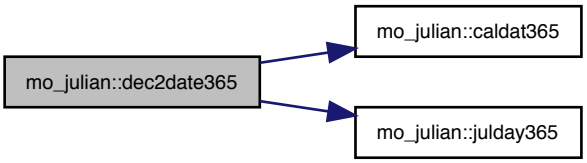
Date

Dec 2015

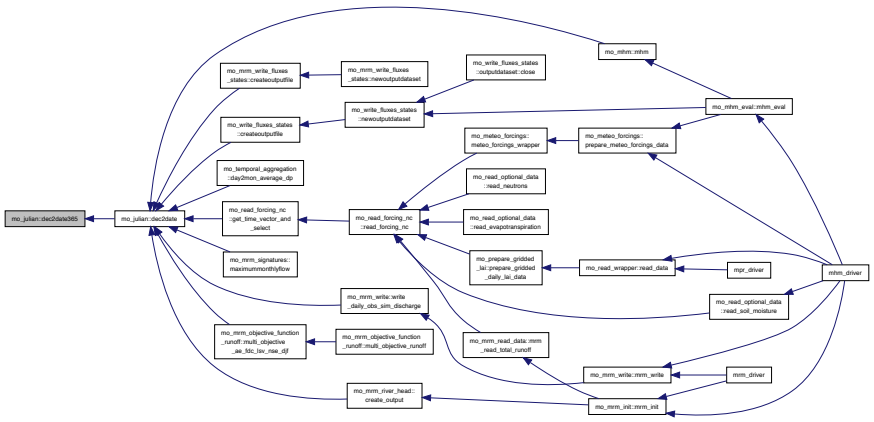
References caldat365(), mo_kind::i4, and julday365().

Referenced by dec2date().

Here is the call graph for this function:



Here is the caller graph for this function:



16.25.2.12 dec2datejulian()

```
elemental subroutine mo_julian::dec2datejulian (
            real(dp), intent(in) julian,
            integer(i4), intent(out), optional dd,
            integer(i4), intent(out), optional mm,
            integer(i4), intent(out), optional yy,
            integer(i4), intent(out), optional hh,
            integer(i4), intent(out), optional nn,
            integer(i4), intent(out), optional ss )  [private]
```

Day, month, year, hour, minute, and second from fractional Julian day.

Inverse of the function date2decJulian. Here dec2dateJulian is input as a fractional Julian Day, which starts at noon of the 1st January 4713 BC, i.e. 01.01.-4712. The routine outputs dd, mm, yy, hh, nn, ss as the day, month, year, hour, minute, and second on which the specified Julian Day started at noon. The zeroth Julian Day is 01.01.-4712 at noon, i.e. the 1st January 4713 BC at noon. Julian day definition starts at 1st January 4713 BC.
Here, the astronomical definition is used, i.e. the year 1 BC (historic) is counted as 0 (astronomic), 2 BC is -1, etc. This means that Julian day definition starts as 01.01.-4712 in astronomical units.

Parameters

in	*real(dp) :: fJulian*	fractional Julian day
out	*integer(i4), optional :: dd*	Day in month of Julian day
out	*integer(i4), optional :: mm*	Month in year of Julian day
out	*integer(i4), optional :: yy*	Year of Julian day
out	*integer(i4), optional :: hh*	Hour of Julian day
out	*integer(i4), optional :: nn*	Minute in hour of Julian day
out	*integer(i4), optional :: ss*	Second in minute of hour of Julian day

Note

Julian day definition starts at noon of the 1st January 4713 BC.
Here, the astronomical definition is used, i.e. the year 1 BC (historic) is counted as 0 (astronomic), 2 BC is -1, etc.

This means that Julian day definition starts as 01.01.-4712 in astronomical units.

juldayJulian and caldatJulian start at midnight of the 1st January 4713 BC. So date2decJulian and juldayJulian as well as dec2dateJulian and caldatJulian are shifted by half a day.
Use date2decJulian with dec2dateJulian together for fractional Julian dates and use juldayJulian with caldat↩ Julian together for integer Julian days.

Author

Written, Matthias Cuntz

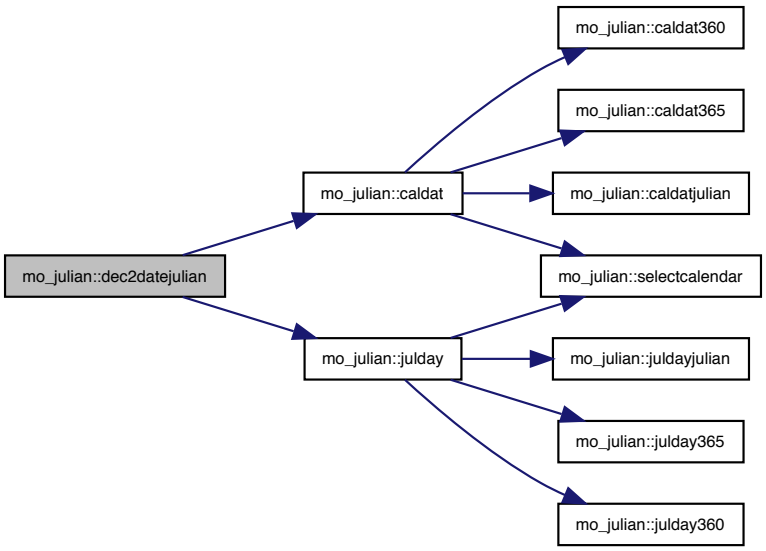
Date

Jan 2013 Modified Matthias Cuntz, May 2014 - changed to new algorithm with astronomical units removed numerical recipes David Schaefer, Jan 2016 - renamed procodure

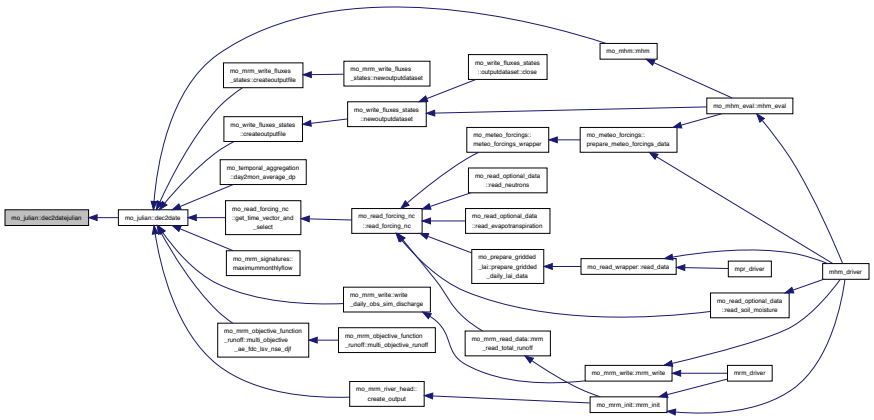
References caldat(), mo_kind::i4, mo_kind::i8, and julday().

Referenced by dec2date().

Here is the call graph for this function:



Here is the caller graph for this function:



16.25.2.13 julday()

```
elemental integer(i4) function, public mo_julian::julday (
             integer(i4), intent(in) dd,
             integer(i4), intent(in) mm,
             integer(i4), intent(in) yy,
             integer(i4), intent(in), optional calendar )
```

Julian day from day, month and year in the current or given calendar.

Wrapper around the calendar specific julday procedures. In this routine julday returns the Julian Day Number that begins at noon of the calendar date specified by month mm, day dd, and year yy, all integer variables. The zeroth Julian Day depends on the called procedure. See their documentation for details.

Parameters

in	*integer(i4) :: dd*	Day in month of Julian day
in	*integer(i4) :: mm*	Month in year of Julian day
in	*integer(i4) :: yy*	Year of Julian day
in	*integer(i4), optional :: calendar*	The calendar to use, the global calendar will be used by default

Returns

integer(i4) :: julian ! Julian day

Author

Written, David Schaefer

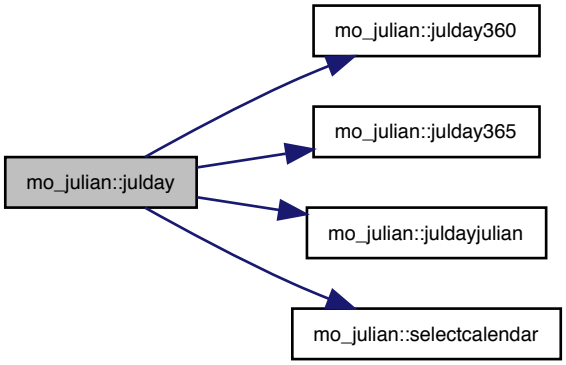
Date

Jan 2015

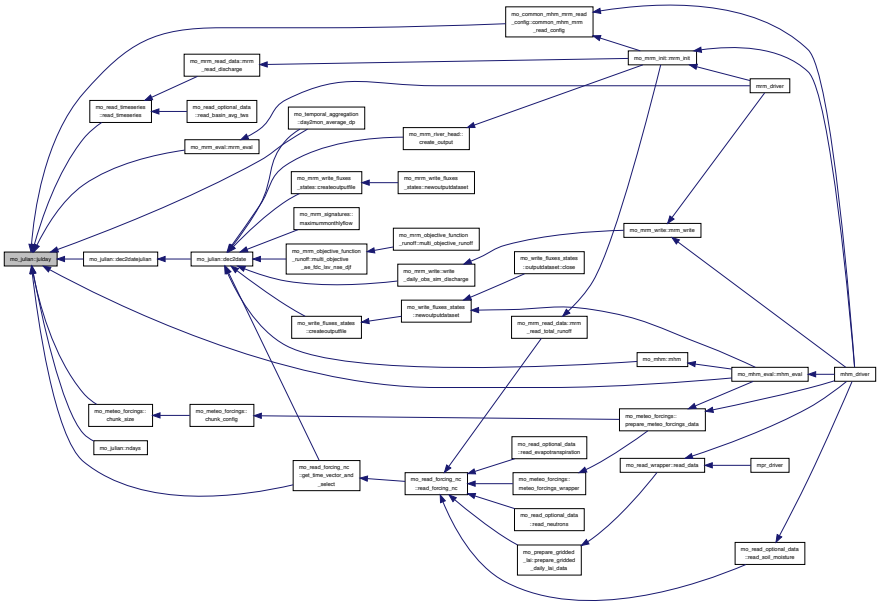
References julday360(), julday365(), juldayjulian(), and selectcalendar().

Referenced by mo_meteo_forcings::chunk_size(), mo_common_mhm_mrm_read_config::common_mhm_mrm_read_config(), mo_temporal_aggregation::day2mon_average_dp(), dec2datejulian(), mo_read_forcing_nc::get_time_vector_and_select(), mo_mhm_eval::mhm_eval(), mo_mrm_eval::mrm_eval(), ndays(), and mo_read_timeseries::read_timeseries().

Here is the call graph for this function:



Here is the caller graph for this function:



16.25.2.14 julday360()

```
elemental integer(i4) function mo_julian::julday360 (
             integer(i4), intent(in) dd,
```

```
            integer(i4), intent(in) mm,
            integer(i4), intent(in) yy )  [private]
```

Julian day from day, month and year in a 360_day calendar.

In this routine julday360 returns the Julian Day Number that begins at noon of the calendar date specified by month mm, day dd, and year yy, all integer variables. The zeroth Julian Day is 01.01.0000

Parameters

in	*integer(i4) :: dd*	Day in month of Julian day
in	*integer(i4) :: mm*	Month in year of Julian day
in	*integer(i4) :: yy*	Year of Julian day

Returns

integer(i4) :: julian ! Julian day

Author

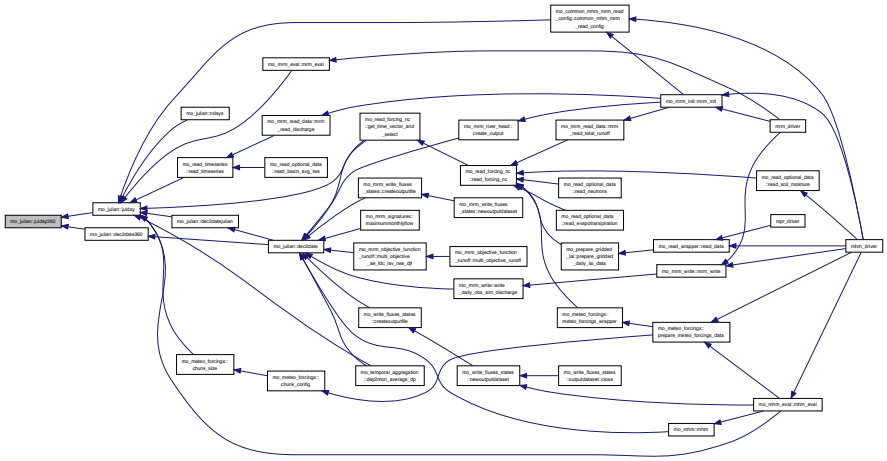
Written, David Schaefer

Date

Oct 2015

Referenced by dec2date360(), and julday().

Here is the caller graph for this function:



16.25.2.15 julday365()

```
elemental integer(i4) function mo_julian::julday365 (
            integer(i4), intent(in) dd,
            integer(i4), intent(in) mm,
            integer(i4), intent(in) yy )  [private]
```

Julian day from day, month and year in a 365_day calendar.

In this routine julday365 returns the Julian Day Number that begins at noon of the calendar date specified by month mm, day dd, and year yy, all integer variables. The zeroth Julian Day is 01.01.0000

Parameters

in	*integer(i4) :: dd*	Day in month of Julian day
in	*integer(i4) :: mm*	Month in year of Julian day
in	*integer(i4) :: yy*	Year of Julian day

Returns

integer(i4) :: julian ! Julian day

Author

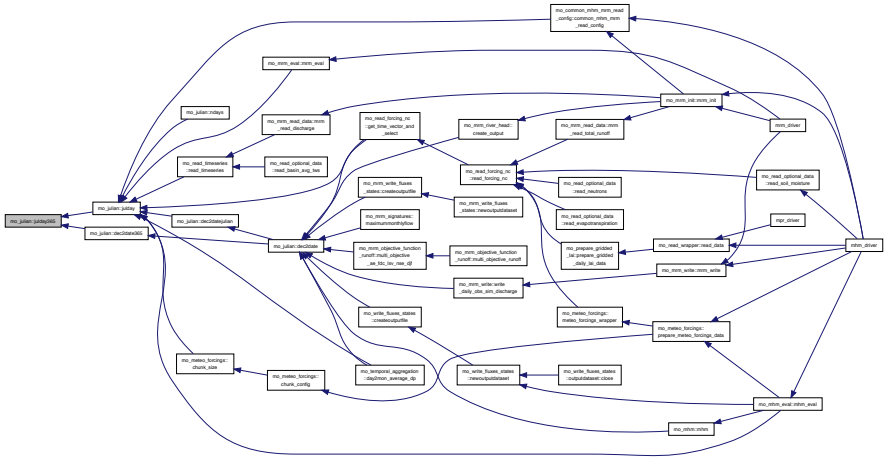
Written, David Schaefer

Date

Dec 2015

Referenced by dec2date365(), and julday().

Here is the caller graph for this function:



16.25.2.16 julday julian()

```
elemental integer(i4) function mo_julian::julday julian (
             integer(i4), intent(in) dd,
             integer(i4), intent(in) mm,
             integer(i4), intent(in) yy )  [private]
```

Julian day from day, month and year.

In this routine julday Julian returns the Julian Day Number that begins at noon of the calendar date specified by month mm, day dd, and year yy, all integer variables. The zeroth Julian Day is 01.01.-4712 at noon, i.e. the 1st January 4713 BC 12:00:00 h. Julian day definition starts at noon of the 1st January 4713 BC.
Here, the astronomical definition is used, i.e. the year 1 BC (historic) is counted as 0 (astronomic), 2 BC is -1, etc. This means that Julian day definition starts as 01.01.-4712 in astronomical units.

Parameters

in	*integer(i4) :: dd*	Day in month of Julian day
in	*integer(i4) :: mm*	Month in year of Julian day
in	*integer(i4) :: yy*	Year of Julian day

Returns

integer(i4) :: julian — Julian day

Note

Julian day definition starts at noon of the 1st January 4713 BC.
Here, the astronomical definition is used, i.e. the year 1 BC (historic) is counted as 0 (astronomic), 2 BC is -1, etc.
This means that Julian day definition starts as 01.01.-4712 in astronomical units.

juldayJulian and caldatJulian start at midnight of the 1st January 4713 BC. So date2decJulian and juldayJulian as well as dec2dateJulian and caldatJulian are shifted by half a day.
Use date2decJulian with dec2dateJulian together for fractional Julian dates and use juldayJulian with caldat↩ Julian together for integer Julian days.

Author

Written, Matthias Cuntz - modified julday from Numerical Recipes

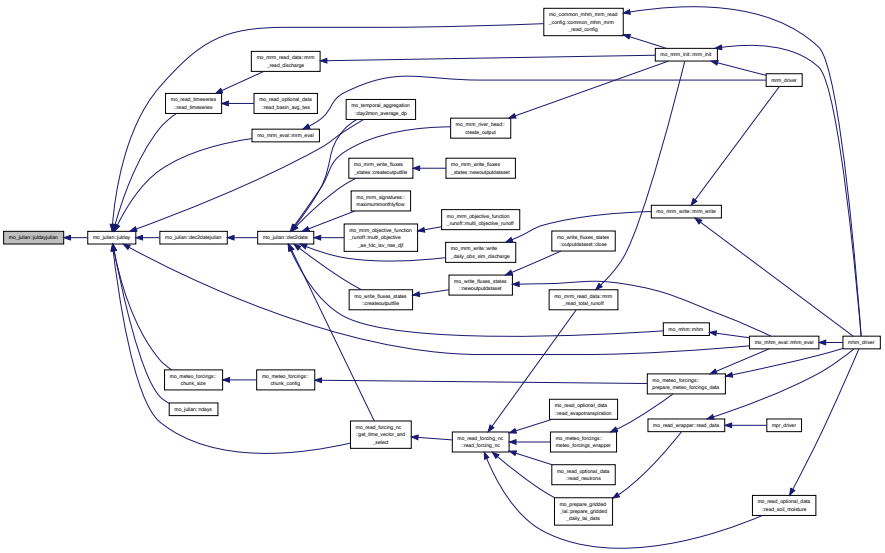
Date

Dec 2011 Modified Matthias Cuntz, May 2014 - changed to new algorithm with astronomical units removed numerical recipes David Schaefer, Jan 2016 - renamed procodure

References mo_kind::i8.

Referenced by julday().

Here is the caller graph for this function:



16.25.2.17 ndays()

```
elemental integer(i4) function, public mo_julian::ndays (
             integer(i4), intent(in) dd,
             integer(i4), intent(in) mm,
             integer(i4), intent(in) yy )
```

IMSL Julian day from day, month and year.

In this routine ndays returns the IMSL Julian Day Number. Julian days begin at noon of the calendar date specified by month mm, day dd, and year yy, all integer variables. IMSL treats 01.01.1900 as a reference and assigns a Julian day 0 to it. ndays = julday(dd,mm,yy) - julday(01,01,1900)

Parameters

in	*integer(i4) :: dd*	Day in month of IMSL Julian day
in	*integer(i4) :: mm*	Month in year of IMSL Julian day
in	*integer(i4) :: yy*	Year of IMSL Julian day

Returns

integer(i4) :: julian — IMSL Julian day, i.e. days before or after 01.01.1900

Author

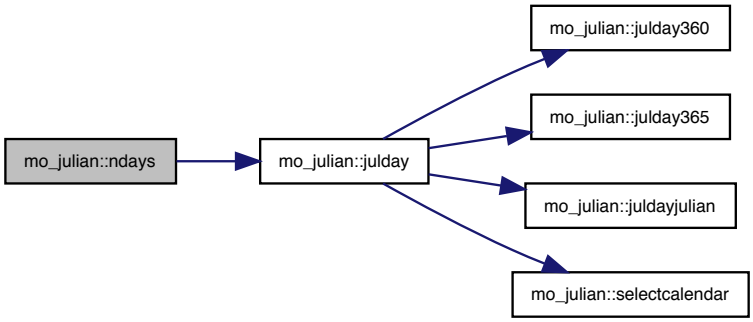
Written, Matthias Cuntz

Date

Dec 2011

References julday().

Here is the call graph for this function:



16.25.2.18 ndyin()

```
elemental subroutine, public mo_julian::ndyin (
             integer(i4), intent(in) julian,
```

```
            integer(i4), intent(out) dd,
            integer(i4), intent(out) mm,
            integer(i4), intent(out) yy )
```

Day, month and year from IMSL Julian day.

Inverse of the function ndys. Here ISML Julian is input as a Julian Day Number minus the Julian Day Number of 01.01.1900, and the routine outputs id, mm, and yy as the day, month, and year on which the specified Julian Day started at noon. ndyin is caldat(IMSLJulian + 2415021, dd, mm, yy)

Parameters

in	*integer(i4) :: julian*	IMSL Julian day, i.e. days before or after 01.01.1900
out	*integer(i4) :: dd*	Day in month of IMSL Julian day
out	*integer(i4) :: mm*	Month in year of IMSL Julian day
out	*integer(i4) :: yy*	Year of IMSL Julian day

Author

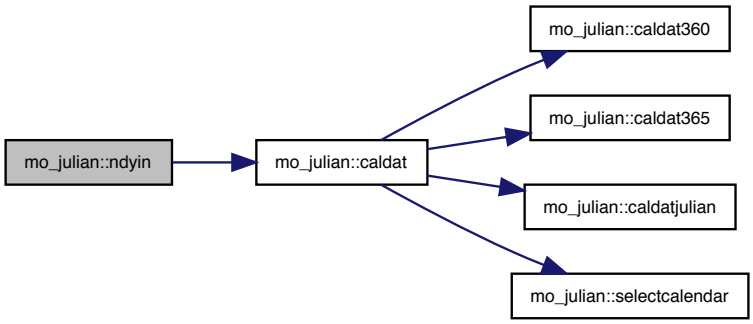
Written, Matthias Cuntz

Date

Dec 2011

References caldat().

Here is the call graph for this function:



16.25.2.19 selectcalendar()

```
pure integer(i4) function mo_julian::selectcalendar (
            integer(i4), intent(in), optional selector )  [private]
```

Select a calendar.

Returns a valid calendar index, based on the given optional argument and/or the module global private variable calendar. If an invalid selector is passed, its value is ignored and the global calendar value retuned instead.

Parameters

in	*integer(i4), optional :: selector*	Calendar selector {1|2|3}

Author

Written, David Schaefer

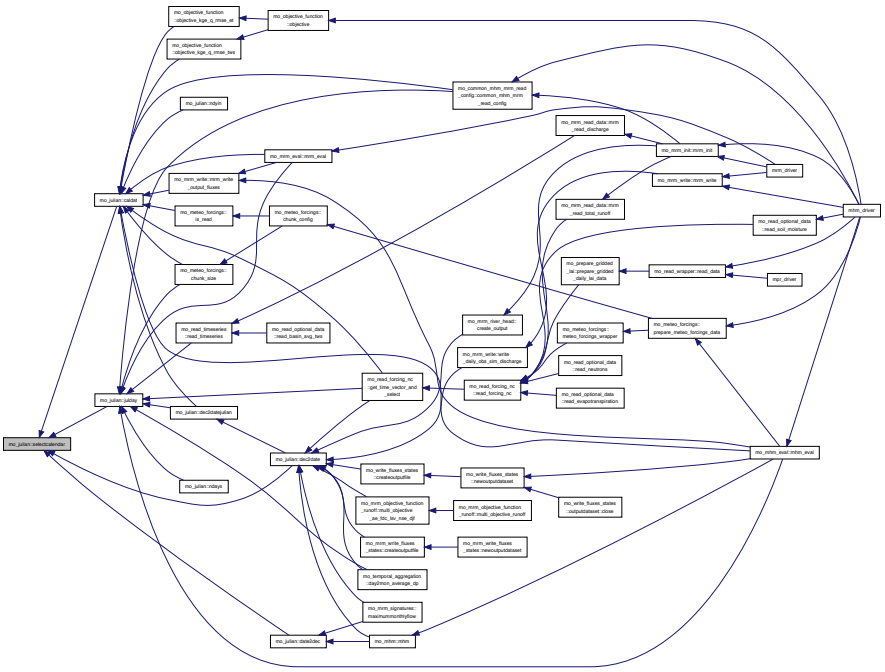
Date

Jan 2015

References calendar.

Referenced by caldat(), date2dec(), dec2date(), and julday().

Here is the caller graph for this function:



16.25.2.20 setcalendarinteger()

```
subroutine mo_julian::setcalendarinteger (
            integer(i4), intent(in) selector )  [private]
```

Set module private variable calendar.

Parameters

in	*integer(i4) :: selector*	{1|2|3}

Author

Written, David Schaefer

Date

Jan 2015

References calendar.

Referenced by setcalendarstring().

Here is the caller graph for this function:

16.25.2.21 setcalendarstring()

```
subroutine mo_julian::setcalendarstring (
             character(*), intent(in) selector )  [private]
```

Set module private variable calendar.

Parameters

in	*character(len=*) :: selector*	{"julian"|"365day"|"360day"}

Author

Written, David Schaefer

Date

Jan 2015

References setcalendarinteger().

Here is the call graph for this function:

16.25.3 Variable Documentation

16.25.3.1 calendar

```
integer(i4), save, private mo_julian::calendar = 1  [private]
```

Referenced by selectcalendar(), and setcalendarinteger().

16.26 mo_kind Module Reference

Define number representations.

Data Types

- type sprs2_dp

  Double Precision Numerical Recipes types for sparse arrays.
- type sprs2_sp

  Single Precision Numerical Recipes types for sparse arrays.

Variables

- integer, parameter i1 = SELECTED_INT_KIND(2)

  1 Byte Integer Kind
- integer, parameter i2 = c_short

  2 Byte Integer Kind
- integer, parameter i4 = c_int

  4 Byte Integer Kind
- integer, parameter i8 = c_long_long

  8 Byte Integer Kind
- integer, parameter sp = c_float

  Single Precision Real Kind.
- integer, parameter dp = c_double

  Double Precision Real Kind.
- integer, parameter spc = c_float_complex

  Single Precision Complex Kind.
- integer, parameter dpc = c_double_complex

  Double Precision Complex Kind.
- integer, parameter lgt = KIND(.true.)

  Logical Kind.

16.26.1 Detailed Description

Define number representations.

This module declares the desired ranges and precisions of the number representations, such as single precision or double precision, 32-bit or 46-bit integer, etc. It confirms mostly with the nrtype module of Numerical Recipes in f90.

Authors

Juliane Mai, Matthias Cuntz, Nov 2011

Date

2011-2014

Copyright

GNU Lesser General Public License http://www.gnu.org/licenses/

16.26.2 Variable Documentation

16.26.2.1 dp

```
integer, parameter mo_kind::dp = c_double
```

Double Precision Real Kind.

Referenced by mo_mrm_routing::add_inflow(), mo_sce::cce(), mo_sce::chkcst(), mo_sce::comp(), mo_anneal::dchange_dp(), mo_dds::dds(), mo_optimization_utils::eval_interface::eval_interface(), mo_read_forcing_nc::get_time_vector_and_select(), mo_sce::getpnt(), mo_anneal::gettemperature_dp(), mo_dds::mdds(), mhm_driver(), mpr_driver(), mrm_driver(), mo_mrm_eval::mrm_eval(), mo_mrm_mpr::mrm_init_param(), mo_mrm_restart::mrm_read_restart_config(), mo_mrm_mpr::mrm_update_param(), mo_mrm_write::mrm_write_output_fluxes(), mo_dds::neigh_value(), mo_optimization_utils::objective_interface::objective_interface(), mo_anneal::pargen_anneal_dp(), mo_mcmc::pargen_dp(), mo_mcmc::pargennorm_dp(), mo_sce::parstt(), mo_spatialsimilarity::pd_dp(), mo_read_forcing_nc::read_const_forcing_nc(), mo_read_forcing_nc::read_forcing_nc(), mo_common_restart::read_grid_info(), mo_restart::read_restart_states(), mo_soil_database::read_soil_lut(), mo_read_forcing_nc::read_weights_nc(), mo_sce::sce(), mo_sce::sort_matrix(), mo_mpr_restart::unpack_field_and_write_1d_dp(), mo_restart::unpack_field_and_write_1d_dp(), mo_mpr_restart::unpack_field_and_write_2d_dp(), mo_restart::unpack_field_and_write_2d_dp(), mo_mpr_restart::unpack_field_and_write_3d_dp(), mo_restart::unpack_field_and_write_3d_dp(), mo_mrm_init::variables_alloc_routing(), mo_mrm_write::write_configfile(), mo_common_restart::write_grid_info(), and mo_restart::write_restart_files().

16.26.2.2 dpc

```
integer, parameter mo_kind::dpc = c_double_complex
```

Double Precision Complex Kind.

Referenced by mo_corr::four1_dp(), mo_corr::four1_sp(), mo_corr::fourrow_dp(), mo_corr::fourrow_sp(), mo_corr::realft_dp(), and mo_corr::zroots_unity_dp().

16.26.2.3 i1

```
integer, parameter mo_kind::i1 = SELECTED_INT_KIND(2)
```

1 Byte Integer Kind

16.26.2.4 i2

```
integer, parameter mo_kind::i2 = c_short
```

2 Byte Integer Kind

16.26.2.5 i4

```
integer, parameter mo_kind::i4 = c_int
```

4 Byte Integer Kind

Referenced by mo_mrm_routing::add_inflow(), mo_julian::caldatjulian(), mo_sce::cce(), mo_sce::chkcst(), mo_↩ meteo_forcings::chunk_config(), mo_meteo_forcings::chunk_size(), mo_sce::comp(), mo_string_utils::compress(), mo_mrm_init::config_output(), mo_anneal::dchange_dp(), mo_dds::dds(), mo_julian::dec2date360(), mo_julian↩ ::dec2date365(), mo_julian::dec2datejulian(), mo_read_forcing_nc::get_time_vector_and_select(), mo_sce↩ ::getpnt(), mo_anneal::gettemperature_dp(), mo_grid::init_lowres_level(), mo_meteo_forcings::is_read(), mo↩ _mrm_init::l0_check_input_routing(), mo_mrm_net_startup::l11_l1_mapping(), mo_neutrons::lookupintegral(), mo_dds::mdds(), mhm_driver(), mo_mhm_eval::mhm_eval(), mo_startup::mhm_initialize(), mo_mpr_startup↩ ::mpr_initialize(), mrm_driver(), mo_mrm_eval::mrm_eval(), mo_mrm_init::mrm_init(), mo_mrm_mpr::mrm_init↩ _param(), mo_mrm_restart::mrm_read_restart_config(), mo_mrm_mpr::mrm_update_param(), mo_mrm_write↩ ::mrm_write_output_fluxes(), mo_optimization::optimization(), mo_anneal::pargen_anneal_dp(), mo_sce::parstt(), mo_percentile::percentile_0d_dp(), mo_percentile::percentile_0d_sp(), mo_percentile::percentile_1d_dp(), mo↩ _percentile::percentile_1d_sp(), mo_mpr_pet::pet_correctbyasp(), mo_mrm_init::print_startup_message(), mo_↩ read_forcing_nc::read_const_forcing_nc(), mo_read_forcing_nc::read_forcing_nc(), mo_common_restart::read_↩ grid_info(), mo_read_lut::read_lai_lut(), mo_restart::read_restart_states(), mo_read_spatial_data::read_spatial_↩ data_ascii_i4(), mo_read_forcing_nc::read_weights_nc(), mo_sce::sce(), mo_sce::sort_matrix(), mo_mpr_restart↩ ::unpack_field_and_write_1d_i4(), mo_restart::unpack_field_and_write_1d_i4(), mo_mpr_restart::unpack_field↩ _and_write_2d_dp(), mo_restart::unpack_field_and_write_2d_dp(), mo_mpr_restart::unpack_field_and_write↩ _3d_dp(), mo_restart::unpack_field_and_write_3d_dp(), mo_upscaling_operators::upscale_arithmetic_mean(), mo_upscaling_operators::upscale_harmonic_mean(), mo_upscaling_operators::upscale_p_norm(), mo_mrm↩ _init::variables_alloc_routing(), mo_write_ascii::write_configfile(), mo_mrm_write::write_configfile(), mo_mpr_↩ restart::write_eff_params(), mo_common_restart::write_grid_info(), mo_mpr_restart::write_mpr_restart_files(), and mo_restart::write_restart_files().

16.26.2.6 i8

```
integer, parameter mo_kind::i8 = c_long_long
```

8 Byte Integer Kind

Referenced by mo_julian::caldatjulian(), mo_sce::cce(), mo_julian::date2decjulian(), mo_anneal::dchange_dp(), mo_dds::dds(), mo_julian::dec2datejulian(), mo_read_forcing_nc::get_time_vector_and_select(), mo_xor4096↩ ::get_timeseed_i8_0d(), mo_xor4096::get_timeseed_i8_1d(), mo_sce::getpnt(), mo_anneal::gettemperature_dp(), mo_julian::juldayjulian(), mo_dds::mdds(), mo_dds::neigh_value(), mo_optimization::optimization(), and mo_sce↩ ::sce().

16.26.2.7 lgt

```
integer, parameter mo_kind::lgt = KIND(.true.)
```

Logical Kind.

16.26.2.8 sp

```
integer, parameter mo_kind::sp = c_float
```

Single Precision Real Kind.

Referenced by mo_spatialsimilarity::pd_sp().

16.26.2.9 spc

```
integer, parameter mo_kind::spc = c_float_complex
```

Single Precision Complex Kind.

Referenced by mo_corr::four1_sp(), mo_corr::fourrow_sp(), mo_corr::realft_sp(), and mo_corr::zroots_unity_sp().

16.27 mo_linfit Module Reference

Fitting a straight line.

Data Types

- interface linfit

  Fits a straight line to input data by minimizing chi^2.

Functions/Subroutines

- real(dp) function, dimension(:), allocatable linfit_dp (x, y, a, b, siga, sigb, chi2, model2)
- real(sp) function, dimension(:), allocatable linfit_sp (x, y, a, b, siga, sigb, chi2, model2)

16.27.1 Detailed Description

Fitting a straight line.

This module provides a routine to fit a straight line with model I or model II regression.

Authors

Matthias Cuntz

Date

Mar 2011

16.27.2 Function/Subroutine Documentation

16.27.2.1 linfit_dp()

```
real(dp) function, dimension(:), allocatable mo_linfit::linfit_dp (
            real(dp), dimension(:), intent(in) x,
            real(dp), dimension(:), intent(in) y,
```

```
            real(dp), intent(out), optional a,
            real(dp), intent(out), optional b,
            real(dp), intent(out), optional siga,
            real(dp), intent(out), optional sigb,
            real(dp), intent(out), optional chi2,
            logical, intent(in), optional model2 )  [private]
```

16.27.2.2 linfit_sp()

```
real(sp) function, dimension(:), allocatable mo_linfit::linfit_sp (
            real(sp), dimension(:), intent(in) x,
            real(sp), dimension(:), intent(in) y,
            real(sp), intent(out), optional a,
            real(sp), intent(out), optional b,
            real(sp), intent(out), optional siga,
            real(sp), intent(out), optional sigb,
            real(sp), intent(out), optional chi2,
            logical, intent(in), optional model2 )  [private]
```

16.28 mo_mad Module Reference

Data Types

- interface mad

Functions/Subroutines

- logical function, dimension(size(arr)) mad_dp (arr, z, mask, deriv)
- logical function, dimension(size(arr)) mad_sp (arr, z, mask, deriv)
- real(dp) function, dimension(size(arr)) mad_val_dp (arr, z, mask, tout, mval)
- real(sp) function, dimension(size(arr)) mad_val_sp (arr, z, mask, tout, mval)

16.28.1 Function/Subroutine Documentation

16.28.1.1 mad_dp()

```
logical function, dimension(size(arr)) mo_mad::mad_dp (
            real(dp), dimension(:), intent(in) arr,
            real(dp), intent(in), optional z,
            logical, dimension(:), intent(in), optional mask,
            integer(i4), intent(in), optional deriv )  [private]
```

16.28.1.2 mad_sp()

```
logical function, dimension(size(arr)) mo_mad::mad_sp (
            real(sp), dimension(:), intent(in) arr,
            real(sp), intent(in), optional z,
```

```
            logical, dimension(:), intent(in), optional mask,
            integer(i4), intent(in), optional deriv )  [private]
```

16.28.1.3 mad_val_dp()

```
real(dp) function, dimension(size(arr)) mo_mad::mad_val_dp (
            real(dp), dimension(:), intent(in) arr,
            real(dp), intent(in), optional z,
            logical, dimension(:), intent(in), optional mask,
            character(1) tout,
            real(dp), intent(in), optional mval )  [private]
```

16.28.1.4 mad_val_sp()

```
real(sp) function, dimension(size(arr)) mo_mad::mad_val_sp (
            real(sp), dimension(:), intent(in) arr,
            real(sp), intent(in), optional z,
            logical, dimension(:), intent(in), optional mask,
            character(1) tout,
            real(sp), intent(in), optional mval )  [private]
```

16.29 mo_mcmc Module Reference

This module is Monte Carlo Markov Chain sampling of a posterior parameter distribution.

Data Types

- interface mcmc

  This module is Monte Carlo Markov Chain sampling of a posterior parameter distribution (measurement errors are either known or modeled).

- interface mcmc_stddev

  This module is Monte Carlo Markov Chain sampling of a posterior parameter distribution (measurement errors are neither known nor modeled).

Functions/Subroutines

- subroutine mcmc_dp (eval, likelihood, para, rangePar, mcmc_paras, burnin_paras, seed_in, printflag_↩ in, maskpara_in, restart, restart_file, tmp_file, loglike_in, ParaSelectMode_in, iter_burnin_in, iter_mcmc_in, chains_in, stepsize_in)
- subroutine mcmc_stddev_dp (eval, likelihood, para, rangePar, mcmc_paras, burnin_paras, seed_in, printflag_in, maskpara_in, tmp_file, loglike_in, ParaSelectMode_in, iter_burnin_in, iter_mcmc_in, chains_in, stepsize_in)
- real(dp) function pargen_dp (old, dMax, oMin, oMax, RN, inbound)
- real(dp) function pargennorm_dp (old, dMax, oMin, oMax, RN, inbound)
- recursive subroutine generatenewparameterset_dp (ParaSelectMode, paraold, truepara, rangePar, stepsize, save_state_2, save_state_3, paranew, ChangePara)

16.29.1 Detailed Description

This module is Monte Carlo Markov Chain sampling of a posterior parameter distribution.

This module is Monte Carlo Markov Chain sampling of a posterior parameter distribution.

Authors

Maren Goehler, Juliane Mai

Date

Aug 2012

16.29.2 Function/Subroutine Documentation

16.29.2.1 generatenewparameterset_dp()

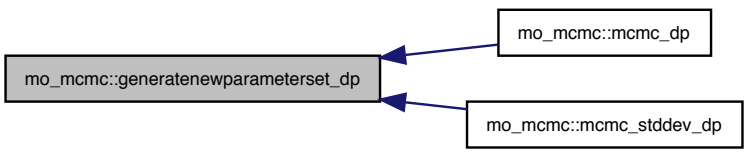
```
recursive subroutine mo_mcmc::generatenewparameterset_dp (
             integer(i4), intent(in) ParaSelectMode,
             real(dp), dimension(:), intent(in) paraold,
             integer(i4), dimension(:), intent(in) truepara,
             real(dp), dimension(:, :), intent(in) rangePar,
             real(dp), dimension(:), intent(in) stepsize,
             integer(i8), dimension(n_save_state), intent(inout) save_state_2,
             integer(i8), dimension(n_save_state), intent(inout) save_state_3,
             real(dp), dimension(size(paraold)), intent(out) paranew,
             logical, dimension(size(paraold)), intent(out) ChangePara )
```

References pargennorm_dp().

Referenced by mcmc_dp(), and mcmc_stddev_dp().

Here is the call graph for this function:

Here is the caller graph for this function:

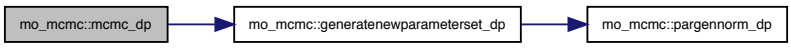


16.29.2.2 mcmc_dp()

```
subroutine mo_mcmc::mcmc_dp (
            procedure(eval_interface), intent(in), pointer eval,
            procedure(objective_interface), intent(in), pointer likelihood,
            real(dp), dimension(:), intent(in) para,
            real(dp), dimension(:, :), intent(in) rangePar,
            real(dp), dimension(:, :), intent(out), allocatable mcmc_paras,
            real(dp), dimension(:, :), intent(out), allocatable burnin_paras,
            integer(i8), intent(in), optional seed_in,
            logical, intent(in), optional printflag_in,
            logical, dimension(size(para, 1)), intent(in), optional maskpara_in,
            logical, intent(in), optional restart,
            character(len = *), intent(in), optional restart_file,
            character(len = *), intent(in), optional tmp_file,
            logical, intent(in), optional loglike_in,
            integer(i4), intent(in), optional ParaSelectMode_in,
            integer(i4), intent(in), optional iter_burnin_in,
            integer(i4), intent(in), optional iter_mcmc_in,
            integer(i4), intent(in), optional chains_in,
            real(dp), dimension(size(para, 1)), intent(in), optional stepsize_in )  [private]
```

References generatenewparameterset_dp(), and mo_xor4096::n_save_state.

Here is the call graph for this function:

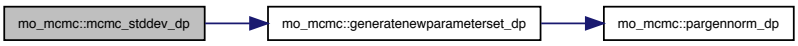


16.29.2.3 mcmc_stddev_dp()

```
subroutine mo_mcmc::mcmc_stddev_dp (
            procedure(eval_interface), intent(in), pointer eval,
            procedure(objective_interface), intent(in), pointer likelihood,
            real(dp), dimension(:), intent(in) para,
            real(dp), dimension(:, :), intent(in) rangePar,
            real(dp), dimension(:, :), intent(out), allocatable mcmc_paras,
            real(dp), dimension(:, :), intent(out), allocatable burnin_paras,
            integer(i8), intent(in), optional seed_in,
            logical, intent(in), optional printflag_in,
            logical, dimension(size(para, 1)), intent(in), optional maskpara_in,
            character(len = *), intent(in), optional tmp_file,
            logical, intent(in), optional loglike_in,
            integer(i4), intent(in), optional ParaSelectMode_in,
            integer(i4), intent(in), optional iter_burnin_in,
            integer(i4), intent(in), optional iter_mcmc_in,
            integer(i4), intent(in), optional chains_in,
            real(dp), dimension(size(para, 1)), intent(in), optional stepsize_in )  [private]
```

References generatenewparameterset_dp(), and mo_xor4096::n_save_state.

Here is the call graph for this function:



16.29.2.4 pargen_dp()

```
real(dp) function mo_mcmc::pargen_dp (
             real(dp), intent(in) old,
             real(dp), intent(in) dMax,
             real(dp), intent(in) oMin,
             real(dp), intent(in) oMax,
             real(dp), intent(in) RN,
             logical, intent(out), optional inbound )  [private]
```

References mo_kind::dp.

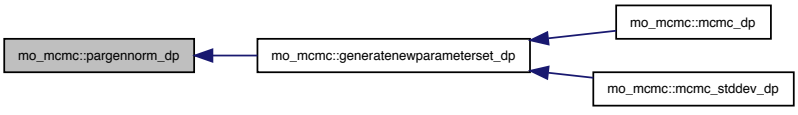
16.29.2.5 pargennorm_dp()

```
real(dp) function mo_mcmc::pargennorm_dp (
             real(dp), intent(in) old,
             real(dp), intent(in) dMax,
             real(dp), intent(in) oMin,
             real(dp), intent(in) oMax,
             real(dp), intent(in) RN,
             logical, intent(out), optional inbound )
```

References mo_kind::dp.

Referenced by generatenewparameterset_dp().

Here is the caller graph for this function:



16.30 mo_message Module Reference

Write out concatenated strings.

Functions/Subroutines

- subroutine, public message (t01, t02, t03, t04, t05, t06, t07, t08, t09, t10, uni, advance)

  Write out several string concatenated either on screen or in a file.

Variables

- character(len=1024), public message_text = ''

16.30.1 Detailed Description

Write out concatenated strings.

Write out several strings concatenated on standard out or a given unit, either advancing or not.

Author

Matthias Cuntz

Date

Jul 2011

16.30.2 Function/Subroutine Documentation

16.30.2.1 message()

```
subroutine, public mo_message::message (
             character(len = *), intent(in), optional t01,
             character(len = *), intent(in), optional t02,
             character(len = *), intent(in), optional t03,
             character(len = *), intent(in), optional t04,
             character(len = *), intent(in), optional t05,
             character(len = *), intent(in), optional t06,
             character(len = *), intent(in), optional t07,
             character(len = *), intent(in), optional t08,
             character(len = *), intent(in), optional t09,
             character(len = *), intent(in), optional t10,
             integer, intent(in), optional uni,
             character(len = *), intent(in), optional advance )
```

Write out several string concatenated either on screen or in a file.

Parameters

in	*character(len=*), optional :: t01*	1st string
in	*character(len=*), optional :: t02*	2nd string
in	*character(len=*), optional :: t03*	3rd string
in	*character(len=*), optional :: t04*	4th string
in	*character(len=*), optional :: t05*	5th string
in	*character(len=*), optional :: t06*	6th string
in	*character(len=*), optional :: t07*	7th string
in	*character(len=*), optional :: t08*	8th string

Parameters

in	*character(len=∗), optional :: t09*	9th string
in	*character(len=∗), optional :: t10*	10th string
in	*integer , optional :: unit*	Unit to write to (default: nout)
in	*character(len=∗), optional :: advance*	WRITE advance keyword (default: 'yes') yes: newline will be written after message no: no newline at end of message

Author

Matthias Cuntz - modified from Echam5, (C) MPI-MET, Hamburg, Germany

Date

Dec 2011

References mo_constants::nout.

Referenced by mo_grid::calculate_grid_properties(), mo_multi_param_reg::canopy_intercept_param(), mo↩_read_wrapper::check_consistency_lut_map(), mo_common_mhm_mrm_read_config::check_optimization_↩settings(), mo_meteo_forcings::chunk_size(), mo_mrm_write_fluxes_states::close(), mo_write_fluxes_states↩::close(), mo_common_mhm_mrm_read_config::common_check_resolution(), mo_common_mhm_mrm_read↩_config::common_mhm_mrm_read_config(), mo_common_read_config::common_read_config(), mo_mrm_init↩::config_output(), mo_startup::constants_init(), mo_mrm_river_head::create_output(), mo_objective_function↩::extract_basin_avg_tws(), mo_mrm_objective_function_runoff::extract_runoff(), mo_soil_database::generate_↩soil_database(), mo_read_forcing_nc::get_time_vector_and_select(), mo_meteo_forcings::is_read(), mo_mpr↩_startup::l0_check_input(), mo_mrm_init::l0_check_input_routing(), mo_mrm_net_startup::l11_flow_direction(), mo_mrm_net_startup::l11_link_location(), mo_startup::l2_variable_init(), mo_mrm_signatures::limb_densities(), mo_mrm_signatures::maximummonthlyflow(), mo_meteo_forcings::meteo_weights_wrapper(), mo_mhm::mhm(), mhm_driver(), mo_mhm_eval::mhm_eval(), mo_mhm_read_config::mhm_read_config(), mo_mrm_signatures↩::moments(), mo_multi_param_reg::mpr(), mo_mpr_eval::mpr_eval(), mo_mpr_read_config::mpr_read_config(), mo_mpr_soilmoist::mpr_sm(), mo_mpr_smhorizons::mpr_smhorizons(), mrm_driver(), mo_mrm_eval::mrm_↩eval(), mo_mrm_init::mrm_init(), mo_mrm_mpr::mrm_init_param(), mo_mrm_read_config::mrm_read_config(), mo_mrm_read_data::mrm_read_discharge(), mo_mrm_read_data::mrm_read_l0_data(), mo_mrm_restart↩::mrm_read_restart_config(), mo_mrm_mpr::mrm_update_param(), mo_mrm_write::mrm_write_optifile(), mo↩_mrm_write::mrm_write_optinamelist(), mo_mrm_restart::mrm_write_restart(), mo_objective_function::objective(), mo_objective_function::objective_et_kge_catchment_avg(), mo_objective_function::objective_kge_q_et(), mo↩_objective_function::objective_kge_q_rmse_et(), mo_objective_function::objective_kge_q_rmse_tws(), mo_↩objective_function::objective_kge_q_sm_corr(), mo_objective_function::objective_neutrons_kge_catchment_↩avg(), mo_objective_function::objective_sm_corr(), mo_objective_function::objective_sm_kge_catchment_avg(), mo_objective_function::objective_sm_pd(), mo_objective_function::objective_sm_sse_standard_score(), mo_↩nml::open_nml(), mo_optimization::optimization(), mo_prepare_gridded_lai::prepare_gridded_daily_lai_data(), mo_prepare_gridded_lai::prepare_gridded_mean_monthly_lai_data(), mo_meteo_forcings::prepare_meteo_↩forcings_data(), mo_mrm_init::print_startup_message(), mo_read_optional_data::read_basin_avg_tws(), mo_↩read_forcing_nc::read_const_forcing_nc(), mo_read_wrapper::read_data(), mo_common_read_data::read_dem(), mo_read_optional_data::read_evapotranspiration(), mo_read_forcing_nc::read_forcing_nc(), mo_common_↩restart::read_grid_info(), mo_read_latlon::read_latlon(), mo_common_read_data::read_lcover(), mo_mrm_read_↩config::read_mrm_routing_params(), mo_read_optional_data::read_neutrons(), mo_soil_database::read_soil_lut(), mo_read_optional_data::read_soil_moisture(), mo_read_timeseries::read_timeseries(), mo_read_forcing_nc↩::read_weights_nc(), mo_mrm_signatures::runoffratio(), mo_common_read_config::set_land_cover_scenes_id(), mo_mrm_objective_function_runoff::single_objective_runoff(), mo_write_ascii::write_configfile(), mo_mrm_write↩::write_configfile(), mo_mrm_write::write_daily_obs_sim_discharge(), mo_mpr_restart::write_mpr_restart_files(), mo_write_ascii::write_optifile(), mo_write_ascii::write_optinamelist(), mo_restart::write_restart_files(), and mo_↩mrm_signatures::zeroflowratio().

16.30.3 Variable Documentation

16.30.3.1 message_text

```
character(len = 1024), public mo_message::message_text = ''
```

Referenced by mo_mpr_startup::l0_check_input(), mo_mrm_init::l0_check_input_routing(), mhm_driver(), mo_nml::open_nml(), mo_nml::position_nml(), and mo_mrm_init::print_startup_message().

16.31 mo_meteo_forcings Module Reference

Prepare meteorological forcings data for mHM.

Functions/Subroutines

- subroutine, public prepare_meteo_forcings_data (iBasin, tt)

  Prepare meteorological forcings data for a given variable.
- subroutine meteo_forcings_wrapper (iBasin, dataPath, inputFormat, dataOut1, lower, upper, ncvarName)

  Prepare meteorological forcings data for mHM at Level-1.
- subroutine meteo_weights_wrapper (iBasin, read_meteo_weights, dataPath, dataOut1, lower, upper, ncvarName)

  Prepare weights for meteorological forcings data for mHM at Level-1.
- subroutine chunk_config (iBasin, tt, read_flag, readPer)

  determines the start date, end date, and read_flag given basin id and current timestep
- logical function is_read (iBasin, tt)

  evaluate whether new chunk should be read at this timestep
- subroutine chunk_size (iBasin, tt, readPer)

  calculate beginning and end of read Period, i.e. that is length of current chunk to read

16.31.1 Detailed Description

Prepare meteorological forcings data for mHM.

Prepare meteorological forcings data for mHM.

Authors

Rohini Kumar

Date

Jan 2012

16.31.2 Function/Subroutine Documentation

16.31.2.1 chunk_config()

```
subroutine mo_meteo_forcings::chunk_config (
            integer(i4), intent(in) iBasin,
            integer(i4), intent(in) tt,
            logical, intent(out) read_flag,
            type(period), intent(out) readPer )
```

determines the start date, end date, and read_flag given basin id and current timestep

TODO: add description

Parameters

in	*integer(i4) :: iBasin*	current Basin
in	*integer(i4) :: tt*	current timestep
out	*logical :: read_flag*	indicate whether reading data should be read
out	*type(period) :: readPer*	start and end dates of reading Period

Authors

Stephan Thober

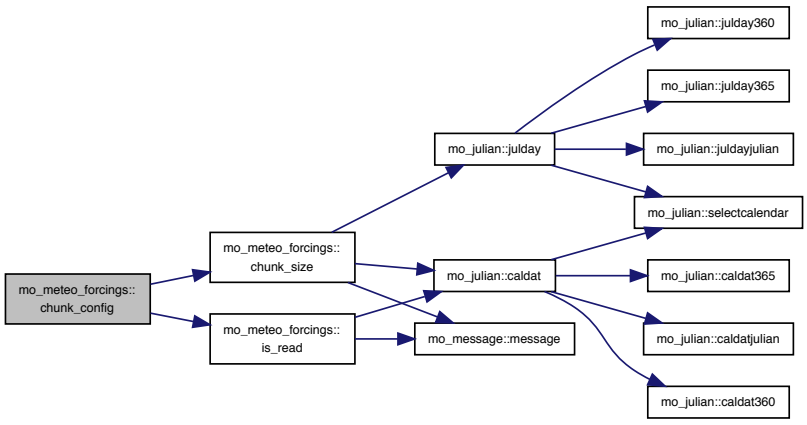
Date

Jun 2014

References chunk_size(), mo_kind::i4, is_read(), and mo_common_constants::nodata_dp.

Referenced by prepare_meteo_forcings_data().

Here is the call graph for this function:



Here is the caller graph for this function:

16.31.2.2 chunk_size()

```
subroutine mo_meteo_forcings::chunk_size (
             integer(i4), intent(in) iBasin,
             integer(i4), intent(in) tt,
             type(period), intent(out) readPer )
```

calculate beginning and end of read Period, i.e. that is length of current chunk to read

TODO: add description

Parameters

in	*integer(i4) :: iBasin*	current Basin to process
in	*integer(i4) :: tt*	current time step
out	*type(period) :: readPer*	start and end dates of read Period

Authors

Stephan Thober

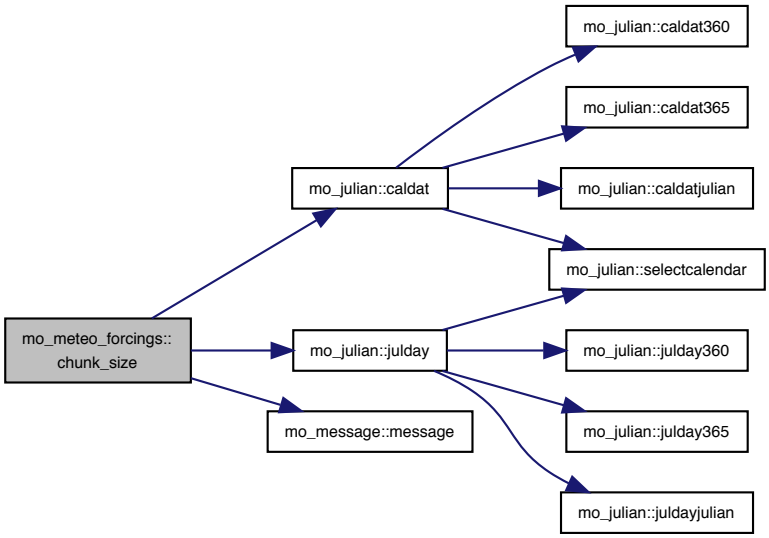
Date

Jun 2014

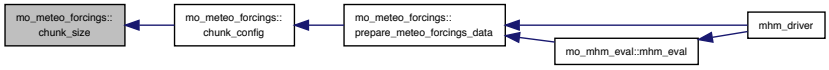
References mo_julian::caldat(), mo_kind::i4, mo_julian::julday(), mo_message::message(), mo_common_mhm↩ _mrm_variables::ntstepday, mo_common_mhm_mrm_variables::simper, and mo_global_variables::timestep_↩ model_inputs.

Referenced by chunk_config().

Here is the call graph for this function:



Here is the caller graph for this function:



16.31.2.3 is_read()

```
logical function mo_meteo_forcings::is_read (
            integer(i4), intent(in) iBasin,
            integer(i4), intent(in) tt )
```

evaluate whether new chunk should be read at this timestep

TODO: add description

Parameters

in	*integer(i4) :: iBasin*	current Basin
in	*integer(i4) :: tt*	current time step

Authors

Stephan Thober

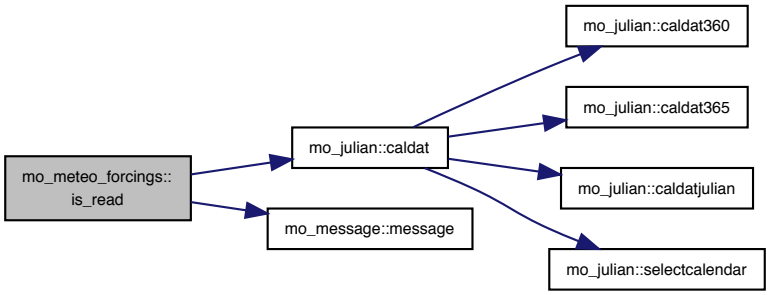
Date

Jun 2014

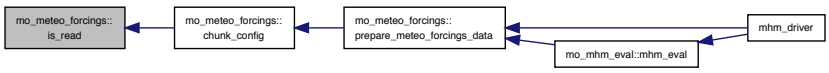
References mo_julian::caldat(), mo_kind::i4, mo_message::message(), mo_common_mhm_mrm_variables↩ ::ntstepday, mo_common_mhm_mrm_variables::simper, mo_common_mhm_mrm_variables::timestep, and mo_↩ global_variables::timestep_model_inputs.

Referenced by chunk_config().

Here is the call graph for this function:



Here is the caller graph for this function:



16.31.2.4 meteo_forcings_wrapper()

```
subroutine mo_meteo_forcings::meteo_forcings_wrapper (
             integer(i4), intent(in) iBasin,
             character(len = *), intent(in) dataPath,
             character(len = *), intent(in) inputFormat,
             real(dp), dimension(:, :), intent(inout), allocatable dataOut1,
             real(dp), intent(in), optional lower,
             real(dp), intent(in), optional upper,
             character(len = *), intent(in), optional ncvarName )
```

Prepare meteorological forcings data for mHM at Level-1.

Prepare meteorological forcings data for mHM, which include 1) Reading meteo. datasets at their native resolution for every basin 2) Perform aggregation or disaggregation of meteo. datasets from their native resolution (level-2) to the required hydrologic resolution (level-1) 3) Pad the above datasets of every basin to their respective global ones

Parameters

in	*integer(i4) :: iBasin*	Basin Id

Parameters

in	*character(len = ∗) :: dataPath*	Data path where a given meteo. variable is stored
in	*character(len = ∗) :: inputFormat*	only 'nc' possible at the moment
in,out	*real(dp), dimension(:, :) :: dataOut1*	Packed meterological variable for the whole simulation period
in	*real(dp), optional :: lower*	Lower bound for check of validity of data values
in	*real(dp), optional :: upper*	Upper bound for check of validity of data values
in	*character(len = ∗), optional :: ncvarName*	name of the variable (for .nc files)

Authors

Rohini Kumar

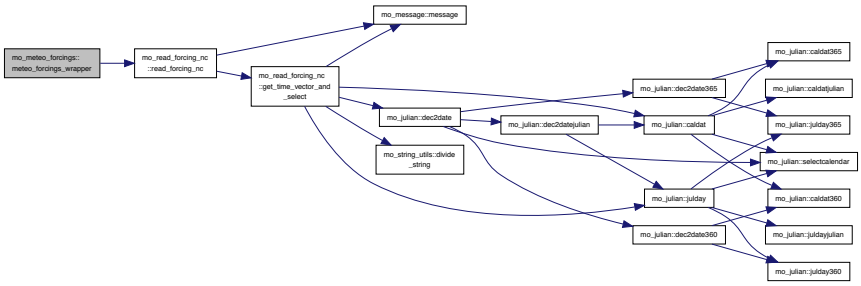
Date

Jan 2013

References mo_common_variables::level1, mo_global_variables::level2, mo_common_constants::nodata_dp, mo_read_forcing_nc::read_forcing_nc(), and mo_common_mhm_mrm_variables::readper.

Referenced by prepare_meteo_forcings_data().

Here is the call graph for this function:



Here is the caller graph for this function:

16.31.2.5 meteo_weights_wrapper()

```
subroutine mo_meteo_forcings::meteo_weights_wrapper (
            integer(i4), intent(in) iBasin,
            logical, intent(in) read_meteo_weights,
```

```
character(len = *), intent(in) dataPath,
real(dp), dimension(:, :, :), intent(inout), allocatable dataOut1,
real(dp), intent(in), optional lower,
real(dp), intent(in), optional upper,
character(len = *), intent(in), optional ncvarName )
```

Prepare weights for meteorological forcings data for mHM at Level-1.

Prepare meteorological weights data for mHM, which include 1) Reading meteo. weights datasets at their native resolution for every basin 2) Perform aggregation or disaggregation of meteo. weights datasets from their native resolution (level-2) to the required hydrologic resolution (level-1) 3) Pad the above datasets of every basin to their respective global ones

Parameters

in	*integer(i4) :: iBasin*	Basin Id
in	*logical :: read_meteo_weights*	Flag for reading meteo weights
in	*character(len = *) :: dataPath*	Data path where a given meteo. variable is stored
in,out	*real(dp), dimension(:, :, :) :: dataOut1*	Packed meterological variable for the whole simulation period
in	*real(dp), optional :: lower*	Lower bound for check of validity of data values
in	*real(dp), optional :: upper*	Upper bound for check of validity of data values
in	*character(len = *), optional :: ncvarName*	name of the variable (for .nc files)

Authors

Stephan Thober & Rohini Kumar

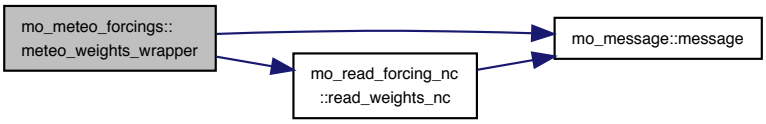
Date

Jan 2017

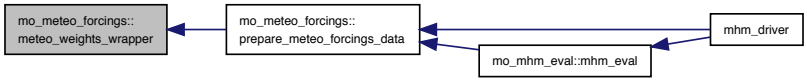
References mo_common_variables::level1, mo_global_variables::level2, mo_message::message(), mo_common_constants::nodata_dp, and mo_read_forcing_nc::read_weights_nc().

Referenced by prepare_meteo_forcings_data().

Here is the call graph for this function:



Here is the caller graph for this function:



16.31.2.6 prepare_meteo_forcings_data()

```
subroutine, public mo_meteo_forcings::prepare_meteo_forcings_data (
            integer(i4), intent(in) iBasin,
            integer(i4), intent(in) tt )
```

Prepare meteorological forcings data for a given variable.

Prepare meteorological forcings data for a given variable. Internally this subroutine calls another routine meteo_↩ wrapper for different meterological variables

Parameters

in	*integer(i4) :: iBasin*	Basin Id
in	*integer(i4) :: tt*	current timestep

Authors

Rohini Kumar

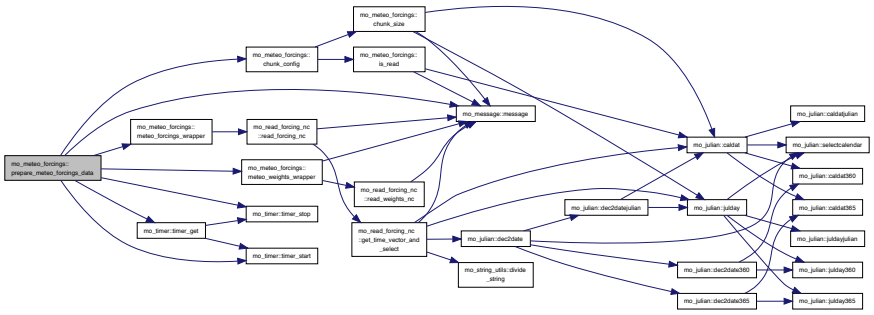
Date

Jan 2013

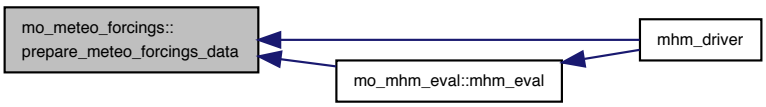
References chunk_config(), mo_global_variables::dirabsvappressure, mo_global_variables::dirmaxtemperature, mo_global_variables::dirmintemperature, mo_global_variables::dirnetradiation, mo_global_variables::dirprecipitation, mo_global_variables::dirreferenceet, mo_global_variables::dirtemperature, mo_global_variables::dirwindspeed, mo_global_variables::inputformat_meteo_forcings, mo_global_variables::l1_absvappress, mo_global_variables↩ ::l1_netrad, mo_global_variables::l1_pet, mo_global_variables::l1_pet_weights, mo_global_variables::l1_pre, mo↩ _global_variables::l1_pre_weights, mo_global_variables::l1_temp, mo_global_variables::l1_temp_weights, mo↩ _global_variables::l1_tmax, mo_global_variables::l1_tmin, mo_global_variables::l1_windspeed, mo_message↩ ::message(), meteo_forcings_wrapper(), meteo_weights_wrapper(), mo_common_variables::nbasins, mo↩ _common_variables::processmatrix, mo_global_variables::read_meteo_weights, mo_common_mhm_mrm_↩ variables::readper, mo_timer::timer_get(), mo_timer::timer_start(), mo_timer::timer_stop(), and mo_global_↩ variables::timestep_model_inputs.

Referenced by mhm_driver(), and mo_mhm_eval::mhm_eval().

Here is the call graph for this function:



Here is the caller graph for this function:



16.32 mo_mhm Module Reference

Call all main processes of mHM.

Functions/Subroutines

- subroutine, public mhm (read_states, tt, time, processMatrix, horizon_depth, nCells1, nHorizons_mHM, ntimesteps_day, c2TSTu, neutron_integral_AFast, global_parameters, latitude, evap_coeff, fday_prec, fnight_prec, fday_pet, fnight_pet, fday_temp, fnight_temp, temp_weights, pet_weights, pre_weights, read_meteo_weights, pet_in, tmin_in, tmax_in, netrad_in, absvappres_in, windspeed_in, prec_in, temp_in, fSealed1, interc, snowpack, sealedStorage, soilMoisture, unsatStorage, satStorage, neutrons, pet_calc, aet_soil, aet_canopy, aet_sealed, baseflow, infiltration, fast_interflow, melt, perc, prec_effect, rain, runoff_sealed, slow_interflow, snow, throughfall, total_runoff, alpha, deg_day_incr, deg_day_max, deg_day_noprec, deg_day, fAsp, petLAIcorFactorL1, HarSamCoeff, PrieTayAlpha, aeroResist, surfResist, frac_roots, interc_max, karst_loss, k0, k1, k2, kp, soil_moist_FC, soil_moist_sat, soil_moist_exponen, jarvis_thresh_c1, temp_thresh, unsat_thresh, water_thresh_sealed, wilting_point)

  Pure mHM calculations.

16.32.1 Detailed Description

Call all main processes of mHM.

This module calls all processes of mHM for a given configuration. The configuration of the model is stored in the a process matrix. This configuration is specified in the namelist mhm.nml. The processes are executed in ascending order. At the moment only

process 5 and 8 have options. Currently the following processes are implemented: Process	Name	Flag	Description
1	interception	1	Maximum interception
2	snow and melting	1	Degree-day
3	soil moisture	1	Feddes equation for ET reduction, Brooks-Corey like
3	soil moisture	2	Jarvis equation for ET reduction, Brooks-Corey like
3	soil moisture	3	Jarvis eq. for ET red. + FC dependency on root frac. coef.
4	direct runoff	1	Linear reservoir exceedance
5	PET	-1	PET is input, LAI based correction, dynamic scaling func.

process 5 and 8 have options. Currently the following processes are implemented: Process	Name	Flag	Description
5	PET	0	PET is input, Aspect based correction
5	PET	1	Hargreaves-Samani
5	PET	2	Priestley-Taylor
5	PET	3	Penman-Monteith
6	interflow	1	Nonlinear reservoir with saturation excess
7	percolation and base flow	1	GW linear reservoir
8	routing	0	no routing
8	routing	1	use mRM i.e. Muskingum
8	routing	2	use mRM i.e. adaptive timestep

Authors

Luis Samaniego

Date

Dec 2012

16.32.2 Function/Subroutine Documentation

16.32.2.1 mhm()

```
subroutine, public mo_mhm::mhm (
            logical, intent(in) read_states,
            integer(i4), intent(in) tt,
            real(dp), intent(in) time,
            integer(i4), dimension(:, :), intent(in) processMatrix,
            real(dp), dimension(:), intent(in) horizon_depth,
            integer(i4), intent(in) nCells1,
            integer(i4), intent(in) nHorizons_mHM,
            real(dp), intent(in) ntimesteps_day,
            real(dp), intent(in) c2TSTu,
            real(dp), dimension(:), intent(in) neutron_integral_AFast,
            real(dp), dimension(:), intent(in) global_parameters,
            real(dp), dimension(:), intent(in) latitude,
            real(dp), dimension(:), intent(in) evap_coeff,
            real(dp), dimension(:), intent(in) fday_prec,
            real(dp), dimension(:), intent(in) fnight_prec,
            real(dp), dimension(:), intent(in) fday_pet,
            real(dp), dimension(:), intent(in) fnight_pet,
            real(dp), dimension(:), intent(in) fday_temp,
            real(dp), dimension(:), intent(in) fnight_temp,
            real(dp), dimension(:, :, :), intent(in) temp_weights,
            real(dp), dimension(:, :, :), intent(in) pet_weights,
            real(dp), dimension(:, :, :), intent(in) pre_weights,
            logical, intent(in) read_meteo_weights,
            real(dp), dimension(:), intent(in) pet_in,
```

```
real(dp), dimension(:), intent(in) tmin_in,
real(dp), dimension(:), intent(in) tmax_in,
real(dp), dimension(:), intent(in) netrad_in,
real(dp), dimension(:), intent(in) absvappres_in,
real(dp), dimension(:), intent(in) windspeed_in,
real(dp), dimension(:), intent(in) prec_in,
real(dp), dimension(:), intent(in) temp_in,
real(dp), dimension(:), intent(inout) fSealed1,
real(dp), dimension(:), intent(inout) interc,
real(dp), dimension(:), intent(inout) snowpack,
real(dp), dimension(:), intent(inout) sealedStorage,
real(dp), dimension(:, :), intent(inout) soilMoisture,
real(dp), dimension(:), intent(inout) unsatStorage,
real(dp), dimension(:), intent(inout) satStorage,
real(dp), dimension(:), intent(inout) neutrons,
real(dp), dimension(:), intent(inout) pet_calc,
real(dp), dimension(:, :), intent(inout) aet_soil,
real(dp), dimension(:), intent(inout) aet_canopy,
real(dp), dimension(:), intent(inout) aet_sealed,
real(dp), dimension(:), intent(inout) baseflow,
real(dp), dimension(:, :), intent(inout) infiltration,
real(dp), dimension(:), intent(inout) fast_interflow,
real(dp), dimension(:), intent(inout) melt,
real(dp), dimension(:), intent(inout) perc,
real(dp), dimension(:), intent(inout) prec_effect,
real(dp), dimension(:), intent(inout) rain,
real(dp), dimension(:), intent(inout) runoff_sealed,
real(dp), dimension(:), intent(inout) slow_interflow,
real(dp), dimension(:), intent(inout) snow,
real(dp), dimension(:), intent(inout) throughfall,
real(dp), dimension(:), intent(inout) total_runoff,
real(dp), dimension(:), intent(inout) alpha,
real(dp), dimension(:), intent(inout) deg_day_incr,
real(dp), dimension(:), intent(inout) deg_day_max,
real(dp), dimension(:), intent(inout) deg_day_noprec,
real(dp), dimension(:), intent(inout) deg_day,
real(dp), dimension(:), intent(inout) fAsp,
real(dp), dimension(:), intent(inout) petLAIcorFactorL1,
real(dp), dimension(:), intent(inout) HarSamCoeff,
real(dp), dimension(:), intent(inout) PrieTayAlpha,
real(dp), dimension(:), intent(inout) aeroResist,
real(dp), dimension(:), intent(inout) surfResist,
real(dp), dimension(:, :), intent(inout) frac_roots,
real(dp), dimension(:), intent(inout) interc_max,
real(dp), dimension(:), intent(inout) karst_loss,
real(dp), dimension(:), intent(inout) k0,
real(dp), dimension(:), intent(inout) k1,
real(dp), dimension(:), intent(inout) k2,
real(dp), dimension(:), intent(inout) kp,
real(dp), dimension(:, :), intent(inout) soil_moist_FC,
real(dp), dimension(:, :), intent(inout) soil_moist_sat,
real(dp), dimension(:, :), intent(inout) soil_moist_exponen,
real(dp), dimension(:), intent(inout) jarvis_thresh_c1,
real(dp), dimension(:), intent(inout) temp_thresh,
real(dp), dimension(:), intent(inout) unsat_thresh,
real(dp), dimension(:), intent(inout) water_thresh_sealed,
real(dp), dimension(:, :), intent(inout) wilting_point )
```

Pure mHM calculations.

Pure mHM calculations. All variables are allocated and initialized. They will be local variables within this call.

Parameters

in	*logical :: read_states*	indicated whether states have been read from file
in	*integer(i4) :: tt*	simulation time step
in	*real(dp) :: time*	current decimal Julian day
in	*integer(i4), dimension(:, :) :: processMatrix*	mHM process configuration matrix
in	*real(dp), dimension(:) :: horizon_depth*	Depth of each horizon in mHM
in	*integer(i4) :: nCells1*	number of cells in a given basin at level L1
in	*integer(i4) :: nHorizons_mHM*	Number of Horizons in mHM
in	*real(dp) :: ntimesteps_day*	number of time intervals per day, transformed in dp
in	*real(dp), dimension(:) :: neutron_integral_AFast*	tabular for neutron flux approximation
in	*real(dp), dimension(:) :: global_parameters*	global mHM parameters
in	*real(dp), dimension(:) :: latitude*	latitude on level 1
in	*real(dp), dimension(:) :: evap_coeff*	Evaporation coefficent for free-water surface of that current month
in	*real(dp), dimension(:) :: fday_prec*	[-] day ratio precipitation < 1
in	*real(dp), dimension(:) :: fnight_prec*	[-] night ratio precipitation < 1
in	*real(dp), dimension(:) :: fday_pet*	[-] day ratio PET < 1
in	*real(dp), dimension(:) :: fnight_pet*	[-] night ratio PET < 1
in	*real(dp), dimension(:) :: fday_temp*	[-] day factor mean temp
in	*real(dp), dimension(:) :: fnight_temp*	[-] night factor mean temp
in	*real(dp), dimension(:, :, :) :: temp_weights*	multiplicative weights for temperature (deg K)
in	*real(dp), dimension(:, :, :) :: pet_weights*	multiplicative weights for potential evapotranspiration
in	*real(dp), dimension(:, :, :) :: pre_weights*	multiplicative weights for precipitation
in	*logical :: read_meteo_weights*	flag whether weights for tavg and pet have read and should be used
in	*real(dp), dimension(:) :: pet_in*	[mm d-1] Daily potential evapotranspiration (input)
in	*real(dp), dimension(:) :: tmin_in*	[degc] Daily minimum temperature
in	*real(dp), dimension(:) :: tmax_in*	[degc] Daily maxumum temperature
in	*real(dp), dimension(:) :: netrad_in*	[w m2] Daily average net radiation
in	*real(dp), dimension(:) :: absvappres_in*	[Pa] Daily average absolute vapour pressure
in	*real(dp), dimension(:) :: windspeed_in*	[m s-1] Daily average wind speed
in	*real(dp), dimension(:) :: prec_in*	[mm d-1] Daily mean precipitation
in	*real(dp), dimension(:) :: temp_in*	[degc] Daily average temperature
in,out	*real(dp), dimension(:) :: fSealed1*	fraction of sealed area at scale L1
in,out	*real(dp), dimension(:) :: interc*	Interception
in,out	*real(dp), dimension(:) :: snowpack*	Snowpack
in,out	*real(dp), dimension(:) :: sealedStorage*	Retention storage of impervious areas
in,out	*real(dp), dimension(:, :) :: soilMoisture*	Soil moisture of each horizon
in,out	*real(dp), dimension(:) :: unsatStorage*	Upper soil storage
in,out	*real(dp), dimension(:) :: satStorage*	Groundwater storage
in,out	*real(dp), dimension(:) :: neutrons*	Ground albedo neutrons
in,out	*real(dp), dimension(:) :: pet_calc*	[mm TST-1] estimated PET (if PET is input = corrected values (fAsp$*$PET))
in,out	*real(dp), dimension(:, :) :: aet_soil*	actual ET

Parameters

in,out	*real(dp), dimension(:) :: aet_canopy*	Real evaporation intensity from canopy
in,out	*real(dp), dimension(:) :: aet_sealed*	Actual ET from free-water surfaces
in,out	*real(dp), dimension(:) :: baseflow*	Baseflow
in,out	*real(dp), dimension(:, :) :: infiltration*	Recharge, infiltration intensity or effective precipitation of each horizon
in,out	*real(dp), dimension(:) :: fast_interflow*	Fast runoff component
in,out	*real(dp), dimension(:) :: melt*	Melting snow depth
in,out	*real(dp), dimension(:) :: perc*	Percolation
in,out	*real(dp), dimension(:) :: prec_effect*	Effective precipitation depth (snow melt + rain)
in,out	*real(dp), dimension(:) :: rain*	Rain precipitation depth
in,out	*real(dp), dimension(:) :: runoff_sealed*	Direct runoff from impervious areas
in,out	*real(dp), dimension(:) :: slow_interflow*	Slow runoff component
in,out	*real(dp), dimension(:) :: snow*	Snow precipitation depth
in,out	*real(dp), dimension(:) :: throughfall*	Throughfall
in,out	*real(dp), dimension(:) :: total_runoff*	Generated runoff
in,out	*real(dp), dimension(:) :: alpha*	Exponent for the upper reservoir
in,out	*real(dp), dimension(:) :: deg_day_incr*	Increase of the Degree-day factor per mm of increase in precipitation
in,out	*real(dp), dimension(:) :: deg_day_max*	Maximum Degree-day factor
in,out	*real(dp), dimension(:) :: deg_day_noprec*	Degree-day factor with no precipitation
in,out	*real(dp), dimension(:) :: deg_day*	Degree-day factor
in,out	*real(dp), dimension(:) :: fAsp*	[1] PET correction for Aspect at level 1
in,out	*real(dp), dimension(:) :: petLAIcorFactorL1*	PET correction factor based on LAI at level 1
in,out	*real(dp), dimension(:) :: HarSamCoeff*	[1] PET Hargreaves Samani coefficient at level 1
in,out	*real(dp), dimension(:) :: PrieTayAlpha*	[1] PET Priestley Taylor coefficient at level 1
in,out	*real(dp), dimension(:) :: aeroResist*	[s m-1] PET aerodynamical resitance at level 1
in,out	*real(dp), dimension(:) :: surfResist*	[s m-1] PET bulk surface resitance at level 1
in,out	*real(dp), dimension(:, :) :: frac_roots*	Fraction of Roots in soil horizon
in,out	*real(dp), dimension(:) :: interc_max*	Maximum interception
in,out	*real(dp), dimension(:) :: karst_loss*	Karstic percolation loss
in,out	*real(dp), dimension(:) :: k0*	Recession coefficient of the upper reservoir, upper outlet
in,out	*real(dp), dimension(:) :: k1*	Recession coefficient of the upper reservoir, lower outlet
in,out	*real(dp), dimension(:) :: k2*	Baseflow recession coefficient
in,out	*real(dp), dimension(:) :: kp*	Percolation coefficient
in,out	*real(dp), dimension(:, :) :: soil_moist_FC*	Soil moisture below which actual ET is reduced
in,out	*real(dp), dimension(:, :) :: soil_moist_sat*	Saturation soil moisture for each horizon [mm]
in,out	*real(dp), dimension(:, :) :: soil_moist_exponen*	Exponential parameter to how non-linear is the soil water retention
in,out	*real(dp), dimension(:) :: jarvis_thresh_c1*	jarvis critical value for normalized soil water content
in,out	*real(dp), dimension(:) :: temp_thresh*	Threshold temperature for snow/rain
in,out	*real(dp), dimension(:) :: unsat_thresh*	Threshold water depth in upper reservoir
in,out	*real(dp), dimension(:) :: water_thresh_sealed*	Threshold water depth in impervious areas
in,out	*real(dp), dimension(:, :) :: wilting_point*	Permanent wilting point for each horizon

Authors

Luis Samaniego & Rohini Kumar

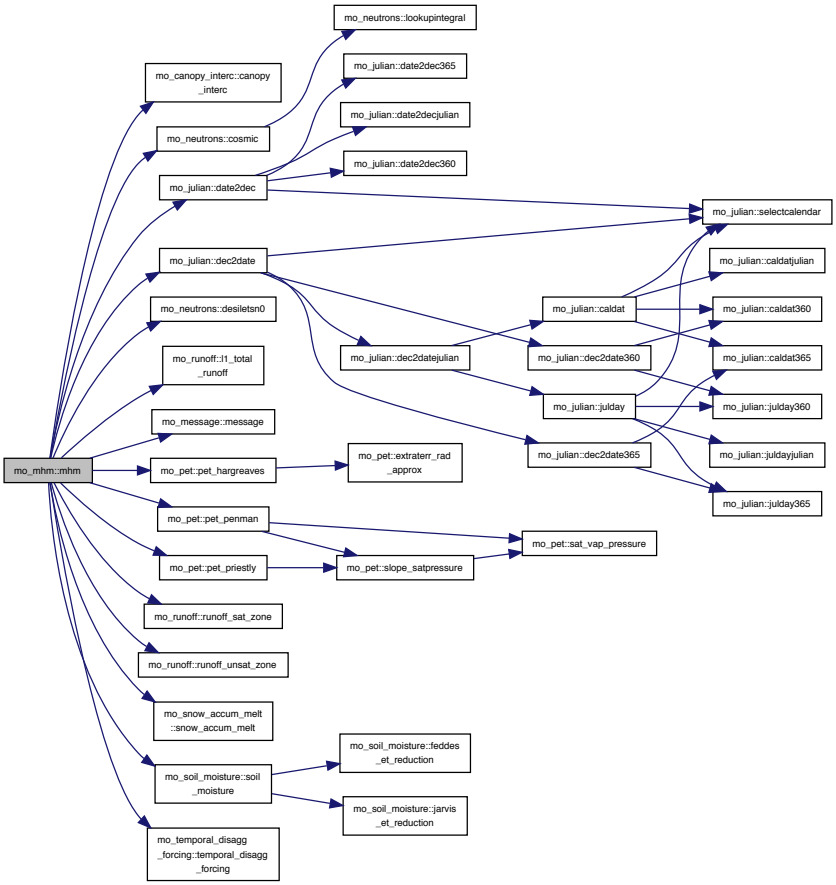
Date

Dec 2012

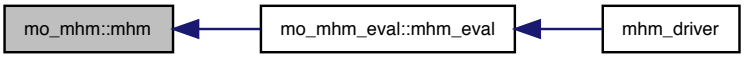
References mo_canopy_interc::canopy_interc(), mo_neutrons::cosmic(), mo_julian::date2dec(), mo_julian::dec2date(), mo_neutrons::desiletsn0(), mo_mhm_constants::harsamconst, mo_runoff::l1_total_runoff(), mo_message::message(), mo_pet::pet_hargreaves(), mo_pet::pet_penman(), mo_pet::pet_priestly(), mo_runoff::runoff_sat_zone(), mo_runoff::runoff_unsat_zone(), mo_snow_accum_melt::snow_accum_melt(), mo_soil_moisture::soil_moisture(), and mo_temporal_disagg_forcing::temporal_disagg_forcing().

Referenced by mo_mhm_eval::mhm_eval().

Here is the call graph for this function:



Here is the caller graph for this function:



16.33 mo_mhm_constants Module Reference

Provides mHM specific constants.

Variables

- real(dp), parameter, public h2odens = 1000.0_dp
- real(dp), parameter, public p2_initstatefluxes = 15.00_dp
- real(dp), parameter, public p3_initstatefluxes = 10.00_dp
- real(dp), parameter, public p4_initstatefluxes = 75.00_dp
- real(dp), parameter, public p5_initstatefluxes = 1500.00_dp
- real(dp), parameter, public c1_initstatesm = 0.25_dp
- integer(i4), parameter, public noutflxstate = 20_i4
- real(dp), parameter, public stboltzmann = 5.67e-08_dp

  Stefan-Boltzmann constant [W m^-2 K^-4].
- real(dp), parameter, public harsamconst = 17.800_dp

  Hargreaves-Samani ref. ET formula [deg C].
- real(dp), parameter, public duffiedr = 0.0330_dp
- real(dp), parameter, public duffiedelta1 = 0.4090_dp
- real(dp), parameter, public duffiedelta2 = 1.3900_dp
- real(dp), parameter, public tetens_c1 = 0.6108_dp

  Tetens's formula to calculate saturated vapour pressure.
- real(dp), parameter, public tetens_c2 = 17.270_dp
- real(dp), parameter, public tetens_c3 = 237.30_dp
- real(dp), parameter, public satpressureslope1 = 4098.0_dp

  calculation of the slope of the saturation vapour pressure curve following Tetens
- real(dp), parameter, public desilets_a0 = 0.0808_dp

  Neutrons and moisture: N0 formula, Desilets et al. 2010.
- real(dp), parameter, public desilets_a1 = 0.372_dp
- real(dp), parameter, public desilets_a2 = 0.115_dp
- real(dp), parameter, public cosmic_bd = 1.4020_dp

  Neutrons and moisture: COSMIC, Shuttleworth et al. 2013.
- real(dp), parameter, public cosmic_vwclat = 0.0753_dp
- real(dp), parameter, public cosmic_n = 348.33_dp
- real(dp), parameter, public cosmic_alpha = 0.2392421548_dp
- real(dp), parameter, public cosmic_l1 = 161.98621864_dp
- real(dp), parameter, public cosmic_l2 = 129.14558985_dp
- real(dp), parameter, public cosmic_l3 = 107.82204562_dp
- real(dp), parameter, public cosmic_l4 = 3.1627190566_dp

16.33.1 Detailed Description

Provides mHM specific constants.

Provides mHM specific constants such as flood plain elevation.

Authors

Matthias Cuntz

Date

Nov 2011

16.33.2 Variable Documentation

16.33.2.1 c1_initstatesm

```
real(dp), parameter, public mo_mhm_constants::c1_initstatesm = 0.25_dp
```

16.33.2.2 cosmic_alpha

```
real(dp), parameter, public mo_mhm_constants::cosmic_alpha = 0.2392421548_dp
```

Referenced by mo_neutrons::cosmic().

16.33.2.3 cosmic_bd

```
real(dp), parameter, public mo_mhm_constants::cosmic_bd = 1.4020_dp
```

Neutrons and moisture: COSMIC, Shuttleworth et al. 2013.

Referenced by mo_neutrons::cosmic().

16.33.2.4 cosmic_l1

```
real(dp), parameter, public mo_mhm_constants::cosmic_l1 = 161.98621864_dp
```

Referenced by mo_neutrons::cosmic().

16.33.2.5 cosmic_l2

```
real(dp), parameter, public mo_mhm_constants::cosmic_l2 = 129.14558985_dp
```

Referenced by mo_neutrons::cosmic().

16.33.2.6 cosmic_l3

```
real(dp), parameter, public mo_mhm_constants::cosmic_l3 = 107.82204562_dp
```

Referenced by mo_neutrons::cosmic().

16.33.2.7 cosmic_l4

```
real(dp), parameter, public mo_mhm_constants::cosmic_l4 = 3.1627190566_dp
```

Referenced by mo_neutrons::cosmic().

16.33.2.8 cosmic_n

```
real(dp), parameter, public mo_mhm_constants::cosmic_n = 348.33_dp
```

Referenced by mo_neutrons::cosmic().

16.33.2.9 cosmic_vwclat

```
real(dp), parameter, public mo_mhm_constants::cosmic_vwclat = 0.0753_dp
```

Referenced by mo_neutrons::cosmic().

16.33.2.10 desilets_a0

```
real(dp), parameter, public mo_mhm_constants::desilets_a0 = 0.0808_dp
```

Neutrons and moisture: N0 formula, Desilets et al. 2010.

Referenced by mo_neutrons::desiletsn0().

16.33.2.11 desilets_a1

```
real(dp), parameter, public mo_mhm_constants::desilets_a1 = 0.372_dp
```

Referenced by mo_neutrons::desiletsn0().

16.33.2.12 desilets_a2

```
real(dp), parameter, public mo_mhm_constants::desilets_a2 = 0.115_dp
```

Referenced by mo_neutrons::desiletsn0().

16.33.2.13 duffiedelta1

```
real(dp), parameter, public mo_mhm_constants::duffiedelta1 = 0.4090_dp
```

Referenced by mo_pet::extraterr_rad_approx().

16.33.2.14 duffiedelta2

```
real(dp), parameter, public mo_mhm_constants::duffiedelta2 = 1.3900_dp
```

Referenced by mo_pet::extraterr_rad_approx().

16.33.2.15 duffiedr

```
real(dp), parameter, public mo_mhm_constants::duffiedr = 0.0330_dp
```

Referenced by mo_pet::extraterr_rad_approx().

16.33.2.16 h2odens

```
real(dp), parameter, public mo_mhm_constants::h2odens = 1000.0_dp
```

Referenced by mo_neutrons::cosmic().

16.33.2.17 harsamconst

```
real(dp), parameter, public mo_mhm_constants::harsamconst = 17.800_dp
```

Hargreaves-Samani ref. ET formula [deg C].

Referenced by mo_mhm::mhm().

16.33.2.18 noutflxstate

```
integer(i4), parameter, public mo_mhm_constants::noutflxstate = 20_i4
```

16.33.2.19 p2_initstatefluxes

```
real(dp), parameter, public mo_mhm_constants::p2_initstatefluxes = 15.00_dp
```

16.33.2.20 p3_initstatefluxes

```
real(dp), parameter, public mo_mhm_constants::p3_initstatefluxes = 10.00_dp
```

16.33.2.21 p4_initstatefluxes

```
real(dp), parameter, public mo_mhm_constants::p4_initstatefluxes = 75.00_dp
```

16.33.2.22 p5_initstatefluxes

```
real(dp), parameter, public mo_mhm_constants::p5_initstatefluxes = 1500.00_dp
```

16.33.2.23 satpressureslope1

```
real(dp), parameter, public mo_mhm_constants::satpressureslope1 = 4098.0_dp
```

calculation of the slope of the saturation vapour pressure curve following Tetens

Referenced by mo_pet::slope_satpressure().

16.33.2.24 stboltzmann

```
real(dp), parameter, public mo_mhm_constants::stboltzmann = 5.67e-08_dp
```

Stefan-Boltzmann constant [W m^-2 K^-4].

16.33.2.25 tetens_c1

```
real(dp), parameter, public mo_mhm_constants::tetens_c1 = 0.6108_dp
```

Tetens's formula to calculate saturated vapour pressure.

Referenced by mo_pet::sat_vap_pressure().

16.33.2.26 tetens_c2

```
real(dp), parameter, public mo_mhm_constants::tetens_c2 = 17.270_dp
```

Referenced by mo_pet::sat_vap_pressure().

16.33.2.27 tetens_c3

```
real(dp), parameter, public mo_mhm_constants::tetens_c3 = 237.30_dp
```

Referenced by mo_pet::sat_vap_pressure(), and mo_pet::slope_satpressure().

16.34 mo_mhm_eval Module Reference

Runs mhm with a specific parameter set and returns required variables, e.g. runoff.

Functions/Subroutines

- subroutine, public mhm_eval (parameterset, runoff, sm_opti, basin_avg_tws, neutrons_opti, et_opti)

  Runs mhm with a specific parameter set and returns required variables, e.g. runoff.

16.34.1 Detailed Description

Runs mhm with a specific parameter set and returns required variables, e.g. runoff.

Runs mhm with a specific parameter set and returns required variables, e.g. runoff.

Authors

Juliane Mai, Rohini Kumar

Date

Feb 2013

16.34.2 Function/Subroutine Documentation

16.34.2.1 mhm_eval()

```
subroutine, public mo_mhm_eval::mhm_eval (
             real(dp), dimension(:), intent(in) parameterset,
             real(dp), dimension(:, :), intent(out), optional, allocatable runoff,
             real(dp), dimension(:, :), intent(out), optional, allocatable sm_opti,
             real(dp), dimension(:, :), intent(out), optional, allocatable basin_avg_tws,
             real(dp), dimension(:, :), intent(out), optional, allocatable neutrons_opti,
             real(dp), dimension(:, :), intent(out), optional, allocatable et_opti )
```

Runs mhm with a specific parameter set and returns required variables, e.g. runoff.

Runs mhm with a specific parameter set and returns required variables, e.g. runoff.

Parameters

in	*real(dp), dimension(:) :: parameterset*	a set of global parameter (gamma) to run mHM, DIMENSION [no. of global_Parameters]
out	*real(dp), dimension(:, :), optional :: runoff*	returns runoff time series, DIMENSION [nTimeSteps, nGaugesTotal]
out	*real(dp), dimension(:, :), optional :: sm_opti*	returns soil moisture time series for all grid cells (of multiple basins concatenated),DIMENSION [nCells, nTimeSteps]
out	*real(dp), dimension(:, :), optional :: basin_avg_tws*	returns basin averaged total water storage time series, DIMENSION [nTimeSteps, nBasins]
out	*real(dp), dimension(:, :), optional :: neutrons_opti*	dim1=ncells, dim2=time
out	*real(dp), dimension(:, :), optional :: et_opti*	returns evapotranspiration time series for all grid cells (of multiple basins concatenated),DIMENSION [nCells, nTimeSteps]

Authors

Juliane Mai, Rohini Kumar

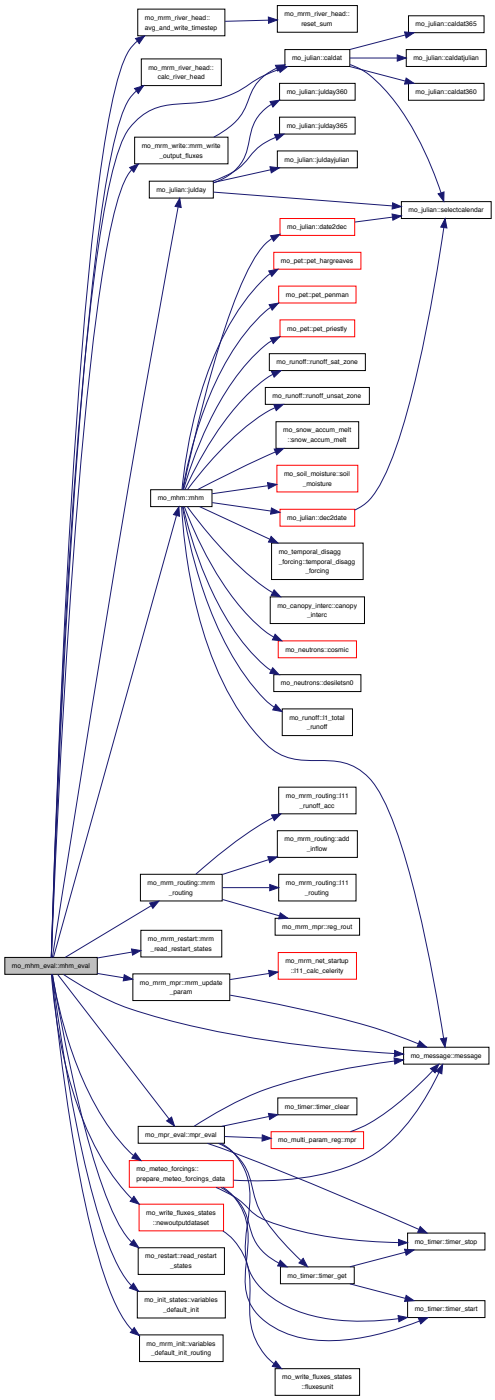
Date

Feb 2013

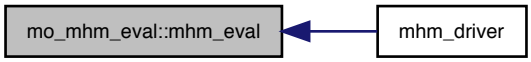
References mo_mrm_river_head::avg_and_write_timestep(), mo_global_variables::basin_avg_tws_sim, mo_mrm_global_variables::basin_mrm, mo_common_mhm_mrm_variables::c2tstu, mo_mrm_river_head::calc_river_head(), mo_julian::caldat(), mo_common_mhm_mrm_variables::dirrestartin, mo_global_variables::evap_coeff, mo_global_variables::fday_pet, mo_global_variables::fday_prec, mo_global_variables::fday_temp, mo_global_variables::fnight_pet, mo_global_variables::fnight_prec, mo_global_variables::fnight_temp, mo_mrm_global_variables::gw_coupling, mo_mpr_global_variables::horizondepth_mhm, mo_common_constants::hoursecs, mo_kind::i4, mo_mrm_global_variables::inflowgauge, mo_julian::julday(), mo_mrm_global_variables::l0_river_head_mon_sum, mo_mrm_global_variables::l11_c1, mo_mrm_global_variables::l11_c2, mo_mrm_global_variables::l11_fromn, mo_mrm_global_variables::l11_l1_id, mo_mrm_global_variables::l11_length, mo_mrm_global_variables::l11_netperm, mo_mrm_global_variables::l11_nlinkfracfpimp, mo_mrm_global_variables::l11_noutlets, mo_mrm_global_variables::l11_qmod, mo_mrm_global_variables::l11_qout, mo_mrm_global_variables::l11_qtin, mo_mrm_global_variables::l11_qtr, mo_mrm_global_variables::l11_slope, mo_mrm_global_variables::l11_ton, mo_mrm_global_variables::l11_tsrout, mo_global_variables::l1_absvappress, mo_mpr_global_variables::l1_aeroresist, mo_global_variables::l1_aetcanopy, mo_global_variables::l1_aetsealed, mo_global_variables::l1_aetsoil, mo_mpr_global_variables::l1_alpha, mo_global_variables::l1_baseflow, mo_mpr_global_variables::l1_degday, mo_mpr_global_variables::l1_degdayinc, mo_mpr_global_variables::l1_degdaymax, mo_mpr_global_variables::l1_degdaynopre, mo_mpr_global_variables::l1_fasp, mo_global_variables::l1_fastrunoff, mo_mpr_global_variables::l1_froots, mo_mpr_global_variables::l1_fsealed, mo_mpr_global_variables::l1_harsamcoeff, mo_global_variables::l1_infilsoil, mo_global_variables::l1_inter, mo_mpr_global_variables::l1_jarvis_thresh_c1, mo_mpr_global_variables::l1_karstloss, mo_mpr_global_variables::l1_kbaseflow, mo_mpr_global_variables::l1_kfastflow, mo_mpr_global_variables::l1_kperco, mo_mpr_global_variables::l1_kslowflow, mo_mrm_global_variables::l1_l11_id, mo_mpr_global_variables::l1_maxinter, mo_global_variables::l1_melt, mo_global_variables::l1_netrad, mo_global_variables::l1_neutrons, mo_global_variables::l1_percol, mo_global_variables::l1_pet, mo_global_variables::l1_pet_calc, mo_global_variables::l1_pet_weights, mo_mpr_global_variables::l1_petlaicorfactor, mo_global_variables::l1_pre, mo_global_variables::l1_pre_weights, mo_global_variables::l1_preeffect, mo_mpr_global_variables::l1_prietayalpha, mo_global_variables::l1_rain, mo_global_variables::l1_runoffseal, mo_global_variables::l1_satstw, mo_mpr_global_variables::l1_sealedthresh, mo_global_variables::l1_sealstw, mo_global_variables::l1_slowrunoff, mo_global_variables::l1_snow, mo_global_variables::l1_snowpack, mo_global_variables::l1_soilmoist, mo_mpr_global_variables::l1_soilmoistexp, mo_mpr_global_variables::l1_soilmoistfc, mo_mpr_global_variables::l1_soilmoistsat, mo_mpr_global_variables::l1_surfresist, mo_global_variables::l1_temp, mo_global_variables::l1_temp_weights, mo_mpr_global_variables::l1_tempthresh, mo_global_variables::l1_throughfall, mo_global_variables::l1_tmax, mo_global_variables::l1_tmin, mo_global_variables::l1_total_runoff, mo_global_variables::l1_unsatstw, mo_mpr_global_variables::l1_unsatthresh, mo_mpr_global_variables::l1_wiltingpoint, mo_global_variables::l1_windspeed, mo_common_mhm_mrm_variables::lcyearid, mo_common_variables::level1, mo_mrm_global_variables::level11, mo_message::message(), mo_mhm::mhm(), mo_mpr_eval::mpr_eval(), mo_mrm_restart::mrm_read_restart_states(), mo_mrm_routing::mrm_routing(), mo_mrm_global_variables::mrm_runoff, mo_mrm_mpr::mrm_update_param(), mo_mrm_write::mrm_write_output_fluxes(), mo_common_variables::nbasins, mo_global_variables::neutron_integral_afast, mo_write_fluxes_states::newoutputdataset(), mo_common_constants::nodata_dp, mo_mpr_global_variables::nsoilhorizons_mhm, mo_global_variables::nsoilhorizons_sm_input, mo_global_variables::ntimesteps_l1_et, mo_global_variables::ntimesteps_l1_neutrons, mo_global_variables::ntimesteps_l1_sm, mo_common_mhm_mrm_variables::ntstepday, mo_common_mhm_mrm_variables::optimize, mo_global_variables::outputflxstate, mo_mrm_global_variables::outputflxstate_mrm, mo_meteo_forcings::prepare_meteo_forcings_data(), mo_common_variables::processmatrix, mo_global_variables::read_meteo_weights, mo_common_mhm_mrm_variables::read_restart, mo_restart::read_restart_states(), mo_common_mhm_mrm_variables::readper, mo_common_variables::resolutionhydrology, mo_common_mhm_mrm_variables::resolutionrouting, mo_common_mhm_mrm_variables::simper, mo_common_mhm_mrm_variables::timestep, mo_global_variables::timestep_et_input, mo_mpr_global_variables::timestep_lai_input, mo_global_variables::timestep_model_inputs, mo_global_variables::timestep_model_outputs, mo_mrm_global_variables::timestep_model_outputs_mrm, mo_global_variables::timestep_sm_input, mo_init_states::variables_default_init(), mo_mrm_init::variables_default_init_routing(), and mo_common_mhm_mrm_variables::warmingdays.

Referenced by mhm_driver().

Here is the call graph for this function:



Here is the caller graph for this function:



16.35 mo_mhm_read_config Module Reference

Reading of main model configurations.

Functions/Subroutines

- subroutine, public mhm_read_config (file_namelist, unamelist)

  Read main configurations for mHM.

16.35.1 Detailed Description

Reading of main model configurations.

This routine reads the configurations of mHM including, input and output directories, module usage specification, simulation time periods, global parameters, ...

Authors

Matthias Zink

Date

Dec 2012

16.35.2 Function/Subroutine Documentation

16.35.2.1 mhm_read_config()

```
subroutine, public mo_mhm_read_config::mhm_read_config (
             character(*), intent(in) file_namelist,
             integer, intent(in) unamelist )
```

Read main configurations for mHM.

The main configurations in mHM are read from three files:

1. mhm.nml
2. mhm_parameters.nml
3. mhm_outputs.nml

For details please refer to the above mentioned namelist files.

Parameters

in	*character(∗) :: file_namelist*	
in	*integer :: unamelist*	

Authors

Matthias Zink

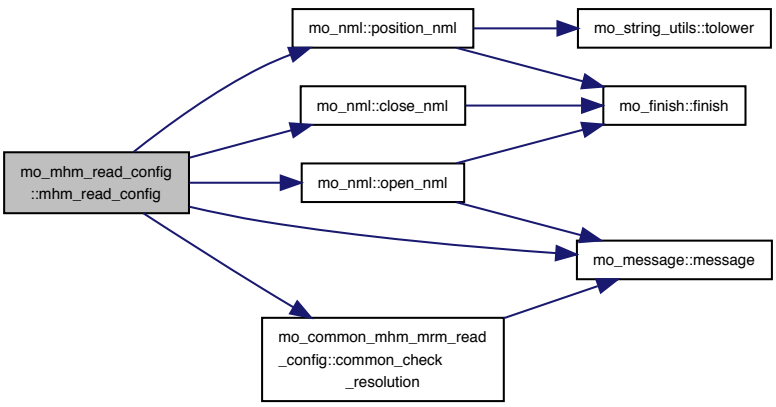
Date

Dec 2012

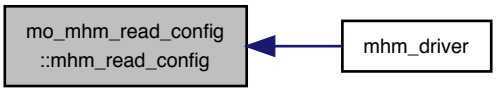
References mo_global_variables::basin_avg_tws_obs, mo_nml::close_nml(), mo_common_mhm_mrm_↩ read_config::common_check_resolution(), mo_global_variables::dirabsvappressure, mo_global_variables↩ ::direvapotranspiration, mo_global_variables::dirmaxtemperature, mo_global_variables::dirmintemperature, mo↩ _global_variables::dirnetradiation, mo_global_variables::dirneutrons, mo_global_variables::dirprecipitation, mo↩ _global_variables::dirreferenceet, mo_global_variables::dirsoil_moisture, mo_global_variables::dirtemperature, mo_global_variables::dirwindspeed, mo_global_variables::evap_coeff, mo_global_variables::fday_pet, mo_↩ global_variables::fday_prec, mo_global_variables::fday_temp, mo_file::file_defoutput, mo_global_variables↩ ::filetws, mo_global_variables::fnight_pet, mo_global_variables::fnight_prec, mo_global_variables::fnight_temp, mo_global_variables::inputformat_meteo_forcings, mo_common_constants::maxnobasins, mo_mpr_constants↩ ::maxnosoilhorizons, mo_message::message(), mo_common_variables::nbasins, mo_common_constants↩ ::nodata_i4, mo_mpr_global_variables::nsoilhorizons_mhm, mo_global_variables::nsoilhorizons_sm_input, mo_↩ nml::open_nml(), mo_common_mhm_mrm_variables::opti_function, mo_common_mhm_mrm_variables::optimize, mo_global_variables::outputflxstate, mo_nml::position_nml(), mo_common_variables::processmatrix, mo_global↩ _variables::read_meteo_weights, mo_global_variables::timestep_et_input, mo_global_variables::timestep_model↩ _inputs, mo_global_variables::timestep_model_outputs, mo_global_variables::timestep_neutrons_input, mo_↩ global_variables::timestep_sm_input, and mo_file::udefoutput.

Referenced by mhm_driver().

Here is the call graph for this function:



Here is the caller graph for this function:



16.36 mo_moment Module Reference

Data Types

- interface absdev
- interface average
- interface central_moment
- interface central_moment_var
- interface correlation
- interface covariance
- interface kurtosis
- interface mean
- interface mixed_central_moment
- interface mixed_central_moment_var
- interface moment
- interface skewness
- interface stddev
- interface variance

Functions/Subroutines

- real(dp) function absdev_dp (dat, mask)
- real(sp) function absdev_sp (dat, mask)
- real(dp) function average_dp (dat, mask)
- real(sp) function average_sp (dat, mask)
- real(dp) function central_moment_dp (x, r, mask)
- real(sp) function central_moment_sp (x, r, mask)
- real(dp) function central_moment_var_dp (x, r, mask)
- real(sp) function central_moment_var_sp (x, r, mask)
- real(dp) function correlation_dp (x, y, mask)
- real(sp) function correlation_sp (x, y, mask)
- real(dp) function covariance_dp (x, y, mask)
- real(sp) function covariance_sp (x, y, mask)
- real(dp) function kurtosis_dp (dat, mask)
- real(sp) function kurtosis_sp (dat, mask)
- real(dp) function mean_dp (dat, mask)
- real(sp) function mean_sp (dat, mask)
- real(dp) function mixed_central_moment_dp (x, y, r, s, mask)
- real(sp) function mixed_central_moment_sp (x, y, r, s, mask)
- real(dp) function mixed_central_moment_var_dp (x, y, r, s, mask)

- real(sp) function mixed_central_moment_var_sp (x, y, r, s, mask)
- subroutine moment_dp (dat, average, variance, skewness, kurtosis, mean, stddev, absdev, mask)
- subroutine moment_sp (dat, average, variance, skewness, kurtosis, mean, stddev, absdev, mask)
- real(dp) function stddev_dp (dat, mask)
- real(sp) function stddev_sp (dat, mask)
- real(dp) function skewness_dp (dat, mask)
- real(sp) function skewness_sp (dat, mask)
- real(dp) function variance_dp (dat, mask)
- real(sp) function variance_sp (dat, mask)

16.36.1 Function/Subroutine Documentation

16.36.1.1 absdev_dp()

```
real(dp) function mo_moment::absdev_dp (
            real(dp), dimension(:), intent(in) dat,
            logical, dimension(:), intent(in), optional mask )  [private]
```

16.36.1.2 absdev_sp()

```
real(sp) function mo_moment::absdev_sp (
            real(sp), dimension(:), intent(in) dat,
            logical, dimension(:), intent(in), optional mask )  [private]
```

16.36.1.3 average_dp()

```
real(dp) function mo_moment::average_dp (
            real(dp), dimension(:), intent(in) dat,
            logical, dimension(:), intent(in), optional mask )  [private]
```

16.36.1.4 average_sp()

```
real(sp) function mo_moment::average_sp (
            real(sp), dimension(:), intent(in) dat,
            logical, dimension(:), intent(in), optional mask )  [private]
```

16.36.1.5 central_moment_dp()

```
real(dp) function mo_moment::central_moment_dp (
            real(dp), dimension(:), intent(in) x,
            integer(i4), intent(in) r,
            logical, dimension(:), intent(in), optional mask )  [private]
```

16.36.1.6 central_moment_sp()

```
real(sp) function mo_moment::central_moment_sp (
             real(sp), dimension(:), intent(in) x,
             integer(i4), intent(in) r,
             logical, dimension(:), intent(in), optional mask )  [private]
```

16.36.1.7 central_moment_var_dp()

```
real(dp) function mo_moment::central_moment_var_dp (
             real(dp), dimension(:), intent(in) x,
             integer(i4), intent(in) r,
             logical, dimension(:), intent(in), optional mask )  [private]
```

16.36.1.8 central_moment_var_sp()

```
real(sp) function mo_moment::central_moment_var_sp (
             real(sp), dimension(:), intent(in) x,
             integer(i4), intent(in) r,
             logical, dimension(:), intent(in), optional mask )  [private]
```

16.36.1.9 correlation_dp()

```
real(dp) function mo_moment::correlation_dp (
             real(dp), dimension(:), intent(in) x,
             real(dp), dimension(:), intent(in) y,
             logical, dimension(:), intent(in), optional mask )  [private]
```

16.36.1.10 correlation_sp()

```
real(sp) function mo_moment::correlation_sp (
             real(sp), dimension(:), intent(in) x,
             real(sp), dimension(:), intent(in) y,
             logical, dimension(:), intent(in), optional mask )  [private]
```

16.36.1.11 covariance_dp()

```
real(dp) function mo_moment::covariance_dp (
             real(dp), dimension(:), intent(in) x,
             real(dp), dimension(:), intent(in) y,
             logical, dimension(:), intent(in), optional mask )  [private]
```

16.36.1.12 covariance_sp()

```
real(sp) function mo_moment::covariance_sp (
```

```
            real(sp), dimension(:), intent(in) x,
            real(sp), dimension(:), intent(in) y,
            logical, dimension(:), intent(in), optional mask )  [private]
```

16.36.1.13 kurtosis_dp()

```
real(dp) function mo_moment::kurtosis_dp (
            real(dp), dimension(:), intent(in) dat,
            logical, dimension(:), intent(in), optional mask )  [private]
```

16.36.1.14 kurtosis_sp()

```
real(sp) function mo_moment::kurtosis_sp (
            real(sp), dimension(:), intent(in) dat,
            logical, dimension(:), intent(in), optional mask )  [private]
```

16.36.1.15 mean_dp()

```
real(dp) function mo_moment::mean_dp (
            real(dp), dimension(:), intent(in) dat,
            logical, dimension(:), intent(in), optional mask )  [private]
```

16.36.1.16 mean_sp()

```
real(sp) function mo_moment::mean_sp (
            real(sp), dimension(:), intent(in) dat,
            logical, dimension(:), intent(in), optional mask )  [private]
```

16.36.1.17 mixed_central_moment_dp()

```
real(dp) function mo_moment::mixed_central_moment_dp (
            real(dp), dimension(:), intent(in) x,
            real(dp), dimension(:), intent(in) y,
            integer(i4), intent(in) r,
            integer(i4), intent(in) s,
            logical, dimension(:), intent(in), optional mask )  [private]
```

16.36.1.18 mixed_central_moment_sp()

```
real(sp) function mo_moment::mixed_central_moment_sp (
            real(sp), dimension(:), intent(in) x,
            real(sp), dimension(:), intent(in) y,
            integer(i4), intent(in) r,
            integer(i4), intent(in) s,
            logical, dimension(:), intent(in), optional mask )  [private]
```

16.36.1.19 mixed_central_moment_var_dp()

```
real(dp) function mo_moment::mixed_central_moment_var_dp (
             real(dp), dimension(:), intent(in) x,
             real(dp), dimension(:), intent(in) y,
             integer(i4), intent(in) r,
             integer(i4), intent(in) s,
             logical, dimension(:), intent(in), optional mask )  [private]
```

16.36.1.20 mixed_central_moment_var_sp()

```
real(sp) function mo_moment::mixed_central_moment_var_sp (
             real(sp), dimension(:), intent(in) x,
             real(sp), dimension(:), intent(in) y,
             integer(i4), intent(in) r,
             integer(i4), intent(in) s,
             logical, dimension(:), intent(in), optional mask )  [private]
```

16.36.1.21 moment_dp()

```
subroutine mo_moment::moment_dp (
             real(dp), dimension(:), intent(in) dat,
             real(dp), intent(out), optional average,
             real(dp), intent(out), optional variance,
             real(dp), intent(out), optional skewness,
             real(dp), intent(out), optional kurtosis,
             real(dp), intent(out), optional mean,
             real(dp), intent(out), optional stddev,
             real(dp), intent(out), optional absdev,
             logical, dimension(:), intent(in), optional mask )  [private]
```

16.36.1.22 moment_sp()

```
subroutine mo_moment::moment_sp (
             real(sp), dimension(:), intent(in) dat,
             real(sp), intent(out), optional average,
             real(sp), intent(out), optional variance,
             real(sp), intent(out), optional skewness,
             real(sp), intent(out), optional kurtosis,
             real(sp), intent(out), optional mean,
             real(sp), intent(out), optional stddev,
             real(sp), intent(out), optional absdev,
             logical, dimension(:), intent(in), optional mask )  [private]
```

16.36.1.23 skewness_dp()

```
real(dp) function mo_moment::skewness_dp (
```

```
            real(dp), dimension(:), intent(in) dat,
            logical, dimension(:), intent(in), optional mask )  [private]
```

16.36.1.24 skewness_sp()

```
real(sp) function mo_moment::skewness_sp (
            real(sp), dimension(:), intent(in) dat,
            logical, dimension(:), intent(in), optional mask )  [private]
```

16.36.1.25 stddev_dp()

```
real(dp) function mo_moment::stddev_dp (
            real(dp), dimension(:), intent(in) dat,
            logical, dimension(:), intent(in), optional mask )  [private]
```

16.36.1.26 stddev_sp()

```
real(sp) function mo_moment::stddev_sp (
            real(sp), dimension(:), intent(in) dat,
            logical, dimension(:), intent(in), optional mask )  [private]
```

16.36.1.27 variance_dp()

```
real(dp) function mo_moment::variance_dp (
            real(dp), dimension(:), intent(in) dat,
            logical, dimension(:), intent(in), optional mask )  [private]
```

16.36.1.28 variance_sp()

```
real(sp) function mo_moment::variance_sp (
            real(sp), dimension(:), intent(in) dat,
            logical, dimension(:), intent(in), optional mask )  [private]
```

16.37 mo_mpr_constants Module Reference

Provides MPR specific constants.

Variables

- integer(i4), parameter, public nlcover_class = 3_i4
- integer(i4), parameter, public maxgeounit = 25_i4
- integer(i4), parameter, public maxnosoilhorizons = 10_i4
- real(dp), parameter, public p2_initstatefluxes = 15.00_dp
- real(dp), parameter, public p3_initstatefluxes = 10.00_dp

- real(dp), parameter, public p4_initstatefluxes = 75.00_dp
- real(dp), parameter, public p5_initstatefluxes = 1500.00_dp
- real(dp), parameter, public c1_initstatesm = 0.25_dp
- real(dp), parameter, public bulkdens_orgmatter = 0.224_dp
- real(dp), parameter, public field_cap_c1 = -0.60_dp
- real(dp), parameter, public field_cap_c2 = 2.0_dp
- real(dp), parameter, public vgenuchten_sandtresh = 66.5_dp
- real(dp), parameter, public vgenuchtenn_c1 = 1.392_dp
- real(dp), parameter, public vgenuchtenn_c2 = 0.418_dp
- real(dp), parameter, public vgenuchtenn_c3 = -0.024_dp
- real(dp), parameter, public vgenuchtenn_c4 = 1.212_dp
- real(dp), parameter, public vgenuchtenn_c5 = -0.704_dp
- real(dp), parameter, public vgenuchtenn_c6 = -0.648_dp
- real(dp), parameter, public vgenuchtenn_c7 = 0.023_dp
- real(dp), parameter, public vgenuchtenn_c8 = 0.044_dp
- real(dp), parameter, public vgenuchtenn_c9 = 3.168_dp
- real(dp), parameter, public vgenuchtenn_c10 = -2.562_dp
- real(dp), parameter, public vgenuchtenn_c11 = 7.0E-9_dp
- real(dp), parameter, public vgenuchtenn_c12 = 4.004_dp
- real(dp), parameter, public vgenuchtenn_c13 = 3.750_dp
- real(dp), parameter, public vgenuchtenn_c14 = -0.016_dp
- real(dp), parameter, public vgenuchtenn_c15 = -4.197_dp
- real(dp), parameter, public vgenuchtenn_c16 = 0.013_dp
- real(dp), parameter, public vgenuchtenn_c17 = 0.076_dp
- real(dp), parameter, public vgenuchtenn_c18 = 0.276_dp
- real(dp), parameter, public ks_c = 10.0_dp
- real(dp), parameter, public pwp_c = 1.0_dp
- real(dp), parameter, public pwp_matpot_thetar = 15000.0_dp
- real(dp), parameter, public windmeasheight = 10.0_dp

  assumed meteorol. measurement hight for estimation of aeroResist and surfResist
- real(dp), parameter, public karman = 0.41_dp

  von karman constant
- real(dp), parameter, public lai_factor_surfresi = 0.3_dp

  LAI factor for bulk surface resistance formulation.
- real(dp), parameter, public lai_offset_surfresi = 1.2_dp

  LAI offset for bulk surface resistance formulation.
- real(dp), parameter, public max_surfresist = 250.0_dp

  maximum bulk surface resistance

16.37.1 Detailed Description

Provides MPR specific constants.

Provides MPR specific constants such as flood plain elevation.

Authors

Matthias Cuntz

Date

Nov 2011

16.37.2 Variable Documentation

16.37.2.1 bulkdens_orgmatter

```
real(dp), parameter, public mo_mpr_constants::bulkdens_orgmatter = 0.224_dp
```

Referenced by mo_mpr_soilmoist::mpr_sm().

16.37.2.2 c1_initstatesm

```
real(dp), parameter, public mo_mpr_constants::c1_initstatesm = 0.25_dp
```

Referenced by mo_init_states::variables_alloc(), and mo_init_states::variables_default_init().

16.37.2.3 field_cap_c1

```
real(dp), parameter, public mo_mpr_constants::field_cap_c1 = -0.60_dp
```

Referenced by mo_mpr_soilmoist::field_cap().

16.37.2.4 field_cap_c2

```
real(dp), parameter, public mo_mpr_constants::field_cap_c2 = 2.0_dp
```

Referenced by mo_mpr_soilmoist::field_cap().

16.37.2.5 karman

```
real(dp), parameter, public mo_mpr_constants::karman = 0.41_dp
```

von karman constant

Referenced by mo_multi_param_reg::aerodynamical_resistance().

16.37.2.6 ks_c

```
real(dp), parameter, public mo_mpr_constants::ks_c = 10.0_dp
```

Referenced by mo_mpr_soilmoist::hydro_cond().

16.37.2.7 lai_factor_surfresi

```
real(dp), parameter, public mo_mpr_constants::lai_factor_surfresi = 0.3_dp
```

LAI factor for bulk surface resistance formulation.

Referenced by mo_mpr_pet::bulksurface_resistance().

16.37.2.8 lai_offset_surfresi

```
real(dp), parameter, public mo_mpr_constants::lai_offset_surfresi = 1.2_dp
```

LAI offset for bulk surface resistance formulation.

Referenced by mo_mpr_pet::bulksurface_resistance().

16.37.2.9 max_surfresist

```
real(dp), parameter, public mo_mpr_constants::max_surfresist = 250.0_dp
```

maximum bulk surface resistance

Referenced by mo_mpr_pet::bulksurface_resistance().

16.37.2.10 maxgeounit

```
integer(i4), parameter, public mo_mpr_constants::maxgeounit = 25_i4
```

Referenced by mo_mpr_read_config::mpr_read_config().

16.37.2.11 maxnosoilhorizons

```
integer(i4), parameter, public mo_mpr_constants::maxnosoilhorizons = 10_i4
```

Referenced by mo_mhm_read_config::mhm_read_config(), and mo_mpr_read_config::mpr_read_config().

16.37.2.12 nlcover_class

```
integer(i4), parameter, public mo_mpr_constants::nlcover_class = 3_i4
```

Referenced by mo_soil_database::read_soil_lut().

16.37.2.13 p2_initstatefluxes

```
real(dp), parameter, public mo_mpr_constants::p2_initstatefluxes = 15.00_dp
```

Referenced by mo_init_states::variables_alloc(), and mo_init_states::variables_default_init().

16.37.2.14 p3_initstatefluxes

```
real(dp), parameter, public mo_mpr_constants::p3_initstatefluxes = 10.00_dp
```

Referenced by mo_init_states::variables_alloc(), and mo_init_states::variables_default_init().

16.37.2.15 p4_initstatefluxes

```
real(dp), parameter, public mo_mpr_constants::p4_initstatefluxes = 75.00_dp
```

Referenced by mo_init_states::variables_alloc(), and mo_init_states::variables_default_init().

16.37.2.16 p5_initstatefluxes

```
real(dp), parameter, public mo_mpr_constants::p5_initstatefluxes = 1500.00_dp
```

Referenced by mo_init_states::variables_alloc(), and mo_init_states::variables_default_init().

16.37.2.17 pwp_c

```
real(dp), parameter, public mo_mpr_constants::pwp_c = 1.0_dp
```

Referenced by mo_mpr_soilmoist::pwp().

16.37.2.18 pwp_matpot_thetar

```
real(dp), parameter, public mo_mpr_constants::pwp_matpot_thetar = 15000.0_dp
```

Referenced by mo_mpr_soilmoist::pwp().

16.37.2.19 vgenuchten_sandtresh

```
real(dp), parameter, public mo_mpr_constants::vgenuchten_sandtresh = 66.5_dp
```

Referenced by mo_mpr_soilmoist::genuchten().

16.37.2.20 vgenuchtenn_c1

```
real(dp), parameter, public mo_mpr_constants::vgenuchtenn_c1 = 1.392_dp
```

Referenced by mo_mpr_soilmoist::genuchten().

16.37.2.21 vgenuchtenn_c10

```
real(dp), parameter, public mo_mpr_constants::vgenuchtenn_c10 = -2.562_dp
```

Referenced by mo_mpr_soilmoist::genuchten().

16.37.2.22 vgenuchtenn_c11

```
real(dp), parameter, public mo_mpr_constants::vgenuchtenn_c11 = 7.0E-9_dp
```

Referenced by mo_mpr_soilmoist::genuchten().

16.37.2.23 vgenuchtenn_c12

```
real(dp), parameter, public mo_mpr_constants::vgenuchtenn_c12 = 4.004_dp
```

Referenced by mo_mpr_soilmoist::genuchten().

16.37.2.24 vgenuchtenn_c13

```
real(dp), parameter, public mo_mpr_constants::vgenuchtenn_c13 = 3.750_dp
```

Referenced by mo_mpr_soilmoist::genuchten().

16.37.2.25 vgenuchtenn_c14

```
real(dp), parameter, public mo_mpr_constants::vgenuchtenn_c14 = -0.016_dp
```

Referenced by mo_mpr_soilmoist::genuchten().

16.37.2.26 vgenuchtenn_c15

```
real(dp), parameter, public mo_mpr_constants::vgenuchtenn_c15 = -4.197_dp
```

Referenced by mo_mpr_soilmoist::genuchten().

16.37.2.27 vgenuchtenn_c16

```
real(dp), parameter, public mo_mpr_constants::vgenuchtenn_c16 = 0.013_dp
```

Referenced by mo_mpr_soilmoist::genuchten().

16.37.2.28 vgenuchtenn_c17

```
real(dp), parameter, public mo_mpr_constants::vgenuchtenn_c17 = 0.076_dp
```

Referenced by mo_mpr_soilmoist::genuchten().

16.37.2.29 vgenuchtenn_c18

```
real(dp), parameter, public mo_mpr_constants::vgenuchtenn_c18 = 0.276_dp
```

Referenced by mo_mpr_soilmoist::genuchten().

16.37.2.30 vgenuchtenn_c2

```
real(dp), parameter, public mo_mpr_constants::vgenuchtenn_c2 = 0.418_dp
```

Referenced by mo_mpr_soilmoist::genuchten().

16.37.2.31 vgenuchtenn_c3

```
real(dp), parameter, public mo_mpr_constants::vgenuchtenn_c3 = -0.024_dp
```

Referenced by mo_mpr_soilmoist::genuchten().

16.37.2.32 vgenuchtenn_c4

```
real(dp), parameter, public mo_mpr_constants::vgenuchtenn_c4 = 1.212_dp
```

Referenced by mo_mpr_soilmoist::genuchten().

16.37.2.33 vgenuchtenn_c5

```
real(dp), parameter, public mo_mpr_constants::vgenuchtenn_c5 = -0.704_dp
```

Referenced by mo_mpr_soilmoist::genuchten().

16.37.2.34 vgenuchtenn_c6

```
real(dp), parameter, public mo_mpr_constants::vgenuchtenn_c6 = -0.648_dp
```

Referenced by mo_mpr_soilmoist::genuchten().

16.37.2.35 vgenuchtenn_c7

```
real(dp), parameter, public mo_mpr_constants::vgenuchtenn_c7 = 0.023_dp
```

Referenced by mo_mpr_soilmoist::genuchten().

16.37.2.36 vgenuchtenn_c8

```
real(dp), parameter, public mo_mpr_constants::vgenuchtenn_c8 = 0.044_dp
```

Referenced by mo_mpr_soilmoist::genuchten().

16.37.2.37 vgenuchtenn_c9

```
real(dp), parameter, public mo_mpr_constants::vgenuchtenn_c9 = 3.168_dp
```

Referenced by mo_mpr_soilmoist::genuchten().

16.37.2.38 windmeasheight

```
real(dp), parameter, public mo_mpr_constants::windmeasheight = 10.0_dp
```

assumed meteorol. measurement hight for estimation of aeroResist and surfResist

Referenced by mo_multi_param_reg::aerodynamical_resistance().

16.38 mo_mpr_eval Module Reference

Runs MPR and writes to global effective parameters.

Functions/Subroutines

- subroutine, public mpr_eval (parameterset)

  Runs MPR and writes to global effective parameters.

16.38.1 Detailed Description

Runs MPR and writes to global effective parameters.

Runs MPR and writes to global effective parameters

Authors

Robert Schweppe

Date

Feb 2018

16.38.2 Function/Subroutine Documentation

16.38.2.1 mpr_eval()

```
subroutine, public mo_mpr_eval::mpr_eval (
             real(dp), dimension(:), intent(in), optional parameterset )
```

Runs MPR and writes to global effective parameters.

Runs MPR and writes to global effective parameters

Parameters

in	*real(dp), dimension(:), optional :: parameterset*	a set of global parameter (gamma) to run mHM, DIMENSION [no. of global_Parameters]

Authors

Juliane Mai, Rohini Kumar

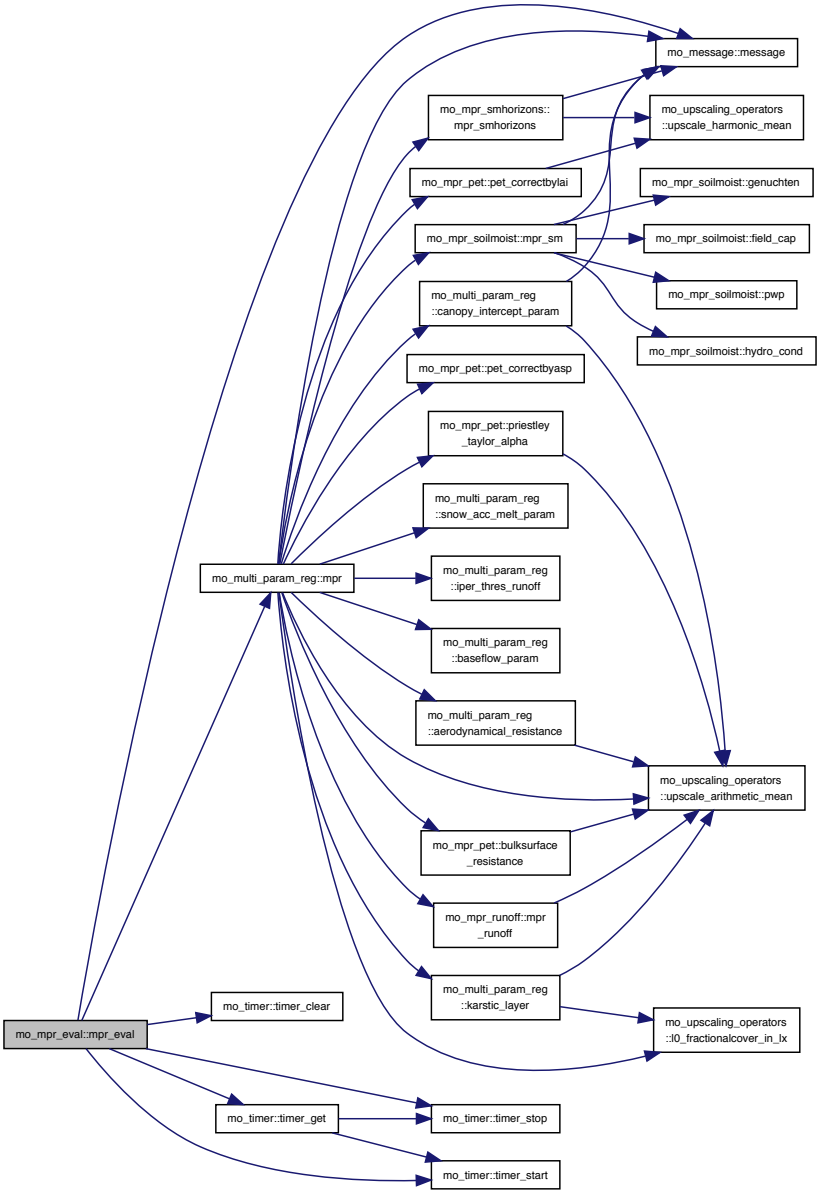
Date

Feb 2013

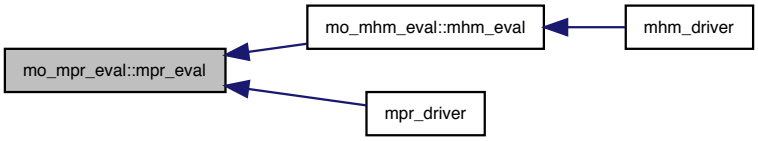
References mo_mpr_global_variables::l0_asp, mo_common_variables::l0_basin, mo_mpr_global_variables::l0_geounit, mo_mpr_global_variables::l0_gridded_lai, mo_common_variables::l0_l1_remap, mo_common_variables::l0_lcover, mo_mpr_global_variables::l0_slope_emp, mo_mpr_global_variables::l0_soilid, mo_mpr_global_variables::l1_aeroresist, mo_mpr_global_variables::l1_alpha, mo_mpr_global_variables::l1_degdayinc, mo_mpr_global_variables::l1_degdaymax, mo_mpr_global_variables::l1_degdaynopre, mo_mpr_global_variables::l1_fasp, mo_mpr_global_variables::l1_froots, mo_mpr_global_variables::l1_fsealed, mo_mpr_global_variables::l1_harsamcoeff, mo_mpr_global_variables::l1_jarvis_thresh_c1, mo_mpr_global_variables::l1_karstloss, mo_mpr_global_variables::l1_kbaseflow, mo_mpr_global_variables::l1_kfastflow, mo_mpr_global_variables::l1_kperco, mo_mpr_global_variables::l1_kslowflow, mo_mpr_global_variables::l1_maxinter, mo_mpr_global_variables::l1_petlaicorfactor, mo_mpr_global_variables::l1_prietayalpha, mo_mpr_global_variables::l1_sealedthresh, mo_mpr_global_variables::l1_soilmoistexp, mo_mpr_global_variables::l1_soilmoistfc, mo_mpr_global_variables::l1_soilmoistsat, mo_mpr_global_variables::l1_surfresist, mo_mpr_global_variables::l1_tempthresh, mo_mpr_global_variables::l1_unsatthresh, mo_mpr_global_variables::l1_wiltingpoint, mo_common_variables::level0, mo_common_variables::level1, mo_message::message(), mo_multi_param_reg::mpr(), mo_common_variables::nbasins, mo_timer::timer_clear(), mo_timer::timer_get(), mo_timer::timer_start(), and mo_timer::timer_stop().

Referenced by mo_mhm_eval::mhm_eval(), and mpr_driver().

Here is the call graph for this function:



Here is the caller graph for this function:



16.39 mo_mpr_file Module Reference

Provides file names and units for mRM.

Variables

- character(len=∗), parameter version = '0.1'

  Current mHM model version.
- character(len=∗), parameter version_date = 'Jun 2019'

  Time of current mHM model version release.
- character(len=∗), parameter file_main = 'mpr_driver.f90'

  Driver file.
- character(len=∗), parameter file_namelist_mpr = 'mpr.nml'

  Namelist file name.
- integer, parameter unamelist_mpr = 80

  Unit for namelist.
- character(len=∗), parameter file_namelist_mpr_param = 'mpr_parameter.nml'

  Parameter namelists file name.
- integer, parameter unamelist_mpr_param = 31

  Unit for namelist.
- character(len=∗), parameter file_soil_database = 'soil_classdefinition.txt'

  Soil database file (iFlag_soilDB = 0) = classical mHM format.
- character(len=∗), parameter file_soil_database_1 = 'soil_classdefinition_iFlag_soilDB_1.txt'

  Soil database file (iFlag_soilDB = 1)
- integer, parameter usoil_database = 52

  Unit for soil data base.
- character(len=∗), parameter file_slope = 'slope.asc'

  slope input data file
- integer, parameter uslope = 54

  Unit for slope input data file.
- character(len=∗), parameter file_aspect = 'aspect.asc'

  aspect input data file
- integer, parameter uaspect = 55

  Unit for aspect input data file.
- character(len=∗), parameter file_hydrogeoclass = 'geology_class.asc'

  hydrogeological classes input data file
- integer, parameter uhydrogeoclass = 58

  Unit for hydrogeological classes input data file.
- character(len=∗), parameter file_soilclass = 'soil_class.asc'

  soil classes input data file
- integer, parameter usoilclass = 59

  Unit for soil classes input data file.
- character(len=∗), parameter file_laiclass = 'LAI_class.asc'

  LAI classes input data file.
- integer, parameter ulaiclass = 60

  Unit for LAI input data file.
- character(len=∗), parameter file_geolut = 'geology_classdefinition.txt'

  geological formation lookup table file
- integer, parameter ugeolut = 64

  Unit for geological formation lookup table file.

- character(len=∗), parameter file_lailut = 'LAI_classdefinition.txt'
  LAI classes lookup table file.
- integer, parameter ulailut = 65
  Unit for LAI classes lookup table file.
- character(len=∗), parameter file_meteo_header = 'header.txt'
  Input nCols and nRows of binary meteo and LAI files are in header file.
- integer, parameter umeteo_header = 50
  Unit for meteo header file.
- character(len=∗), parameter file_meteo_binary_end = '.bin'
  File ending of meteo files.
- integer, parameter umeteo = 51
  Unit for meteo files.

16.39.1 Detailed Description

Provides file names and units for mRM.

Provides all filenames as well as all units used for the multiscale Routing Model mRM.

Authors

Matthias Cuntz, Stephan Thober

Date

Aug 2015

16.39.2 Variable Documentation

16.39.2.1 file_aspect

```
character(len = *), parameter mo_mpr_file::file_aspect = 'aspect.asc'
```

aspect input data file

Referenced by mo_read_wrapper::read_data().

16.39.2.2 file_geolut

```
character(len = *), parameter mo_mpr_file::file_geolut = 'geology_classdefinition.txt'
```

geological formation lookup table file

Referenced by mo_read_wrapper::read_data().

16.39.2.3 file_hydrogeoclass

```
character(len = *), parameter mo_mpr_file::file_hydrogeoclass = 'geology_class.asc'
```

hydrogeological classes input data file

Referenced by mo_startup::constants_init(), and mo_read_wrapper::read_data().

16.39.2.4 file_laiclass

```
character(len = *), parameter mo_mpr_file::file_laiclass = 'LAI_class.asc'
```

LAI classes input data file.

Referenced by mo_read_wrapper::read_data().

16.39.2.5 file_lailut

```
character(len = *), parameter mo_mpr_file::file_lailut = 'LAI_classdefinition.txt'
```

LAI classes lookup table file.

Referenced by mo_read_wrapper::read_data().

16.39.2.6 file_main

```
character(len = *), parameter mo_mpr_file::file_main = 'mpr_driver.f90'
```

Driver file.

16.39.2.7 file_meteo_binary_end

```
character(len = *), parameter mo_mpr_file::file_meteo_binary_end = '.bin'
```

File ending of meteo files.

16.39.2.8 file_meteo_header

```
character(len = *), parameter mo_mpr_file::file_meteo_header = 'header.txt'
```

Input nCols and nRows of binary meteo and LAI files are in header file.

Referenced by mo_startup::l2_variable_init().

16.39.2.9 file_namelist_mpr

```
character(len = *), parameter mo_mpr_file::file_namelist_mpr = 'mpr.nml'
```

Namelist file name.

16.39.2.10 file_namelist_mpr_param

```
character(len = *), parameter mo_mpr_file::file_namelist_mpr_param = 'mpr_parameter.nml'
```

Parameter namelists file name.

Referenced by mpr_driver().

16.39.2.11 file_slope

```
character(len = *), parameter mo_mpr_file::file_slope = 'slope.asc'
```

slope input data file

Referenced by mo_mrm_read_data::mrm_read_l0_data(), and mo_read_wrapper::read_data().

16.39.2.12 file_soil_database

```
character(len = *), parameter mo_mpr_file::file_soil_database = 'soil_classdefinition.txt'
```

Soil database file (iFlag_soilDB = 0) = classical mHM format.

Referenced by mo_read_wrapper::read_data().

16.39.2.13 file_soil_database_1

```
character(len = *), parameter mo_mpr_file::file_soil_database_1 = 'soil_classdefinition_iFlag↩
_soilDB_1.txt'
```

Soil database file (iFlag_soilDB = 1)

Referenced by mo_read_wrapper::read_data().

16.39.2.14 file_soilclass

```
character(len = *), parameter mo_mpr_file::file_soilclass = 'soil_class.asc'
```

soil classes input data file

Referenced by mo_read_wrapper::read_data().

16.39.2.15 uaspect

```
integer, parameter mo_mpr_file::uaspect = 55
```

Unit for aspect input data file.

Referenced by mo_read_wrapper::read_data().

16.39.2.16 ugeolut

```
integer, parameter mo_mpr_file::ugeolut = 64
```

Unit for geological formation lookup table file.

Referenced by mo_read_wrapper::read_data().

16.39.2.17 uhydrogeoclass

```
integer, parameter mo_mpr_file::uhydrogeoclass = 58
```

Unit for hydrogeological classes input data file.

Referenced by mo_read_wrapper::read_data().

16.39.2.18 ulaiclass

```
integer, parameter mo_mpr_file::ulaiclass = 60
```

Unit for LAI input data file.

Referenced by mo_read_wrapper::read_data().

16.39.2.19 ulailut

```
integer, parameter mo_mpr_file::ulailut = 65
```

Unit for LAI classes lookup table file.

Referenced by mo_read_wrapper::read_data().

16.39.2.20 umeteo

```
integer, parameter mo_mpr_file::umeteo = 51
```

Unit for meteo files.

16.39.2.21 umeteo_header

```
integer, parameter mo_mpr_file::umeteo_header = 50
```

Unit for meteo header file.

Referenced by mo_startup::l2_variable_init().

16.39.2.22 unamelist_mpr

```
integer, parameter mo_mpr_file::unamelist_mpr = 80
```

Unit for namelist.

16.39.2.23 unamelist_mpr_param

```
integer, parameter mo_mpr_file::unamelist_mpr_param = 31
```

Unit for namelist.

Referenced by mpr_driver().

16.39.2.24 uslope

```
integer, parameter mo_mpr_file::uslope = 54
```

Unit for slope input data file.

Referenced by mo_mrm_read_data::mrm_read_l0_data(), and mo_read_wrapper::read_data().

16.39.2.25 usoil_database

```
integer, parameter mo_mpr_file::usoil_database = 52
```

Unit for soil data base.

Referenced by mo_soil_database::read_soil_lut().

16.39.2.26 usoilclass

```
integer, parameter mo_mpr_file::usoilclass = 59
```

Unit for soil classes input data file.

Referenced by mo_read_wrapper::read_data().

16.39.2.27 version

```
character(len = *), parameter mo_mpr_file::version = '0.1'
```

Current mHM model version.

16.39.2.28 version_date

```
character(len = *), parameter mo_mpr_file::version_date = 'Jun 2019'
```

Time of current mHM model version release.

16.40 mo_mpr_global_variables Module Reference

Global variables for mpr only.

Data Types

- type soiltype

Variables

- real(dp), public tillagedepth

- integer(i4), public nsoiltypes
- integer(i4), public iflag_soildb
- integer(i4), public nsoilhorizons_mhm
- real(dp), dimension(:), allocatable, public horizondepth_mhm
- type(soiltype), public soildb
- integer(i4), public ngeounits
- integer(i4), dimension(:), allocatable, public geounitlist
- integer(i4), dimension(:), allocatable, public geounitkar
- character(256), public inputformat_gridded_lai
- integer(i4), public timestep_lai_input
- integer(i4), public nlaiclass
- integer(i4), public nlai
- integer(i4), dimension(:), allocatable, public laiunitlist
- real(dp), dimension(:, :), allocatable, public lailut
- type(period), dimension(:), allocatable, public laiper
- real(dp), public fracsealed_cityarea
- real(dp), dimension(:), allocatable, public l0_slope_emp
- real(dp), dimension(:, :), allocatable, public l0_gridded_lai
- real(dp), dimension(:), allocatable, public l0_slope
- real(dp), dimension(:), allocatable, public l0_asp
- integer(i4), dimension(:, :), allocatable, public l0_soilid
- integer(i4), dimension(:), allocatable, public l0_geounit
- character(256), dimension(:), allocatable, public dirgridded_lai
- real(dp), dimension(:, :, :), allocatable, public l1_fsealed
- real(dp), dimension(:, :, :), allocatable, public l1_alpha
- real(dp), dimension(:, :, :), allocatable, public l1_degdayinc
- real(dp), dimension(:, :, :), allocatable, public l1_degdaymax
- real(dp), dimension(:, :, :), allocatable, public l1_degdaynopre
- real(dp), dimension(:, :, :), allocatable, public l1_degday
- real(dp), dimension(:, :, :), allocatable, public l1_karstloss
- real(dp), dimension(:, :, :), allocatable, public l1_fasp
- real(dp), dimension(:, :, :), allocatable, public l1_petlaicorfactor
- real(dp), dimension(:, :, :), allocatable, public l1_harsamcoeff
- real(dp), dimension(:, :, :), allocatable, public l1_prietayalpha
- real(dp), dimension(:, :, :), allocatable, public l1_aeroresist
- real(dp), dimension(:, :, :), allocatable, public l1_surfresist
- real(dp), dimension(:, :, :), allocatable, public l1_froots
- real(dp), dimension(:, :, :), allocatable, public l1_maxinter
- real(dp), dimension(:, :, :), allocatable, public l1_kfastflow
- real(dp), dimension(:, :, :), allocatable, public l1_kslowflow
- real(dp), dimension(:, :, :), allocatable, public l1_kbaseflow
- real(dp), dimension(:, :, :), allocatable, public l1_kperco
- real(dp), dimension(:, :, :), allocatable, public l1_soilmoistfc
- real(dp), dimension(:, :, :), allocatable, public l1_soilmoistsat
- real(dp), dimension(:, :, :), allocatable, public l1_soilmoistexp
- real(dp), dimension(:, :, :), allocatable, public l1_jarvis_thresh_c1
- real(dp), dimension(:, :, :), allocatable, public l1_tempthresh
- real(dp), dimension(:, :, :), allocatable, public l1_unsatthresh
- real(dp), dimension(:, :, :), allocatable, public l1_sealedthresh
- real(dp), dimension(:, :, :), allocatable, public l1_wiltingpoint

16.40.1 Detailed Description

Global variables for mpr only.

TODO: add description

Authors

Robert Schweppe

Date

Dec 2017

16.40.2 Variable Documentation

16.40.2.1 dirgridded_lai

```
character(256), dimension(:), allocatable, public mo_mpr_global_variables::dirgridded_lai
```

Referenced by mo_mpr_read_config::mpr_read_config(), mo_prepare_gridded_lai::prepare_gridded_daily_lai_↩data(), and mo_prepare_gridded_lai::prepare_gridded_mean_monthly_lai_data().

16.40.2.2 fracsealed_cityarea

```
real(dp), public mo_mpr_global_variables::fracsealed_cityarea
```

Referenced by mo_multi_param_reg::mpr(), and mo_mpr_read_config::mpr_read_config().

16.40.2.3 geounitkar

```
integer(i4), dimension(:), allocatable, public mo_mpr_global_variables::geounitkar
```

Referenced by mo_multi_param_reg::karstic_layer(), and mo_read_wrapper::read_data().

16.40.2.4 geounitlist

```
integer(i4), dimension(:), allocatable, public mo_mpr_global_variables::geounitlist
```

Referenced by mo_multi_param_reg::baseflow_param(), mo_startup::constants_init(), mo_multi_param_reg↩::karstic_layer(), and mo_read_wrapper::read_data().

16.40.2.5 horizondepth_mhm

```
real(dp), dimension(:), allocatable, public mo_mpr_global_variables::horizondepth_mhm
```

Referenced by mo_soil_database::generate_soil_database(), mo_mhm_eval::mhm_eval(), mo_multi_param_reg↩::mpr(), mo_mpr_read_config::mpr_read_config(), mo_soil_database::read_soil_lut(), mo_init_states::variables_↩alloc(), and mo_init_states::variables_default_init().

16.40.2.6 iflag_soildb

```
integer(i4), public mo_mpr_global_variables::iflag_soildb
```

Referenced by mo_soil_database::generate_soil_database(), mo_mpr_startup::l0_check_input(), mo_mpr_↩ startup::l0_variable_init(), mo_multi_param_reg::mpr(), mo_mpr_read_config::mpr_read_config(), mo_mpr_↩ soilmoist::mpr_sm(), mo_read_wrapper::read_data(), and mo_soil_database::read_soil_lut().

16.40.2.7 inputformat_gridded_lai

```
character(256), public mo_mpr_global_variables::inputformat_gridded_lai
```

Referenced by mo_mpr_read_config::mpr_read_config(), and mo_prepare_gridded_lai::prepare_gridded_daily_↩ lai_data().

16.40.2.8 l0_asp

```
real(dp), dimension(:), allocatable, public mo_mpr_global_variables::l0_asp
```

Referenced by mo_mpr_startup::l0_check_input(), mo_mpr_eval::mpr_eval(), and mo_read_wrapper::read_data().

16.40.2.9 l0_geounit

```
integer(i4), dimension(:), allocatable, public mo_mpr_global_variables::l0_geounit
```

Referenced by mo_mpr_startup::l0_check_input(), mo_mpr_eval::mpr_eval(), and mo_read_wrapper::read_data().

16.40.2.10 l0_gridded_lai

```
real(dp), dimension(:, :), allocatable, public mo_mpr_global_variables::l0_gridded_lai
```

Referenced by mo_mpr_startup::l0_check_input(), mo_mpr_eval::mpr_eval(), mo_prepare_gridded_lai::prepare↩ _gridded_daily_lai_data(), mo_prepare_gridded_lai::prepare_gridded_mean_monthly_lai_data(), and mo_read_↩ wrapper::read_data().

16.40.2.11 l0_slope

```
real(dp), dimension(:), allocatable, public mo_mpr_global_variables::l0_slope
```

Referenced by mo_mpr_startup::l0_check_input(), mo_mpr_startup::l0_variable_init(), mo_mrm_net_startup::l11↩ _calc_celerity(), mo_mrm_read_data::mrm_read_l0_data(), mo_mrm_restart::mrm_read_restart_config(), mo_↩ mrm_restart::mrm_write_restart(), and mo_read_wrapper::read_data().

16.40.2.12 l0_slope_emp

```
real(dp), dimension(:), allocatable, public mo_mpr_global_variables::l0_slope_emp
```

Referenced by mo_mpr_startup::l0_variable_init(), and mo_mpr_eval::mpr_eval().

16.40.2.13 l0_soilid

```
integer(i4), dimension(:, :), allocatable, public mo_mpr_global_variables::l0_soilid
```

Referenced by mo_mpr_startup::l0_check_input(), mo_mpr_startup::l0_variable_init(), mo_mpr_eval::mpr_eval(), and mo_read_wrapper::read_data().

16.40.2.14 l1_aeroresist

```
real(dp), dimension(:, :, :), allocatable, public mo_mpr_global_variables::l1_aeroresist
```

Referenced by mo_mpr_startup::init_eff_params(), mo_mhm_eval::mhm_eval(), mo_mpr_eval::mpr_eval(), mo_↩ restart::read_restart_states(), mo_init_states::variables_default_init(), and mo_mpr_restart::write_eff_params().

16.40.2.15 l1_alpha

```
real(dp), dimension(:, :, :), allocatable, public mo_mpr_global_variables::l1_alpha
```

Referenced by mo_mpr_startup::init_eff_params(), mo_mhm_eval::mhm_eval(), mo_mpr_eval::mpr_eval(), mo_↩ restart::read_restart_states(), mo_init_states::variables_default_init(), and mo_mpr_restart::write_eff_params().

16.40.2.16 l1_degday

```
real(dp), dimension(:, :, :), allocatable, public mo_mpr_global_variables::l1_degday
```

Referenced by mo_mpr_startup::init_eff_params(), mo_mhm_eval::mhm_eval(), mo_restart::read_restart_states(), mo_init_states::variables_default_init(), and mo_mpr_restart::write_eff_params().

16.40.2.17 l1_degdayinc

```
real(dp), dimension(:, :, :), allocatable, public mo_mpr_global_variables::l1_degdayinc
```

Referenced by mo_mpr_startup::init_eff_params(), mo_mhm_eval::mhm_eval(), mo_mpr_eval::mpr_eval(), mo_↩ restart::read_restart_states(), mo_init_states::variables_default_init(), and mo_mpr_restart::write_eff_params().

16.40.2.18 l1_degdaymax

```
real(dp), dimension(:, :, :), allocatable, public mo_mpr_global_variables::l1_degdaymax
```

Referenced by mo_mpr_startup::init_eff_params(), mo_mhm_eval::mhm_eval(), mo_mpr_eval::mpr_eval(), mo_↩ restart::read_restart_states(), mo_init_states::variables_default_init(), and mo_mpr_restart::write_eff_params().

16.40.2.19 l1_degdaynopre

```
real(dp), dimension(:, :, :), allocatable, public mo_mpr_global_variables::l1_degdaynopre
```

Referenced by mo_mpr_startup::init_eff_params(), mo_mhm_eval::mhm_eval(), mo_mpr_eval::mpr_eval(), mo_↩ restart::read_restart_states(), mo_init_states::variables_default_init(), and mo_mpr_restart::write_eff_params().

16.40.2.20 l1_fasp

```
real(dp), dimension(:, :, :), allocatable, public mo_mpr_global_variables::l1_fasp
```

Referenced by mo_mpr_startup::init_eff_params(), mo_mhm_eval::mhm_eval(), mo_mpr_eval::mpr_eval(), mo_↩ restart::read_restart_states(), mo_init_states::variables_default_init(), and mo_mpr_restart::write_eff_params().

16.40.2.21 l1_froots

```
real(dp), dimension(:, :, :), allocatable, public mo_mpr_global_variables::l1_froots
```

Referenced by mo_mpr_startup::init_eff_params(), mo_mhm_eval::mhm_eval(), mo_mpr_eval::mpr_eval(), mo_↩ restart::read_restart_states(), mo_init_states::variables_default_init(), and mo_mpr_restart::write_eff_params().

16.40.2.22 l1_fsealed

```
real(dp), dimension(:, :, :), allocatable, public mo_mpr_global_variables::l1_fsealed
```

Referenced by mo_mpr_startup::init_eff_params(), mo_mhm_eval::mhm_eval(), mo_mpr_eval::mpr_eval(), mo_↩ restart::read_restart_states(), mo_init_states::variables_default_init(), and mo_mpr_restart::write_eff_params().

16.40.2.23 l1_harsamcoeff

```
real(dp), dimension(:, :, :), allocatable, public mo_mpr_global_variables::l1_harsamcoeff
```

Referenced by mo_mpr_startup::init_eff_params(), mo_mhm_eval::mhm_eval(), mo_mpr_eval::mpr_eval(), mo_↩ restart::read_restart_states(), mo_init_states::variables_default_init(), and mo_mpr_restart::write_eff_params().

16.40.2.24 l1_jarvis_thresh_c1

```
real(dp), dimension(:, :, :), allocatable, public mo_mpr_global_variables::l1_jarvis_thresh_c1
```

Referenced by mo_mpr_startup::init_eff_params(), mo_mhm_eval::mhm_eval(), mo_mpr_eval::mpr_eval(), mo_↩ restart::read_restart_states(), mo_init_states::variables_default_init(), and mo_mpr_restart::write_eff_params().

16.40.2.25 l1_karstloss

```
real(dp), dimension(:, :, :), allocatable, public mo_mpr_global_variables::l1_karstloss
```

Referenced by mo_mpr_startup::init_eff_params(), mo_mhm_eval::mhm_eval(), mo_mpr_eval::mpr_eval(), mo_↩ restart::read_restart_states(), mo_init_states::variables_default_init(), and mo_mpr_restart::write_eff_params().

16.40.2.26 l1_kbaseflow

```
real(dp), dimension(:, :, :), allocatable, public mo_mpr_global_variables::l1_kbaseflow
```

Referenced by mo_mpr_startup::init_eff_params(), mo_mhm_eval::mhm_eval(), mo_mpr_eval::mpr_eval(), mo_↩ restart::read_restart_states(), mo_init_states::variables_default_init(), and mo_mpr_restart::write_eff_params().

16.40.2.27 l1_kfastflow

```
real(dp), dimension(:, :, :), allocatable, public mo_mpr_global_variables::l1_kfastflow
```

Referenced by mo_mpr_startup::init_eff_params(), mo_mhm_eval::mhm_eval(), mo_mpr_eval::mpr_eval(), mo_↩ restart::read_restart_states(), mo_init_states::variables_default_init(), and mo_mpr_restart::write_eff_params().

16.40.2.28 l1_kperco

```
real(dp), dimension(:, :, :), allocatable, public mo_mpr_global_variables::l1_kperco
```

Referenced by mo_mpr_startup::init_eff_params(), mo_mhm_eval::mhm_eval(), mo_mpr_eval::mpr_eval(), mo_↩ restart::read_restart_states(), mo_init_states::variables_default_init(), and mo_mpr_restart::write_eff_params().

16.40.2.29 l1_kslowflow

```
real(dp), dimension(:, :, :), allocatable, public mo_mpr_global_variables::l1_kslowflow
```

Referenced by mo_mpr_startup::init_eff_params(), mo_mhm_eval::mhm_eval(), mo_mpr_eval::mpr_eval(), mo_↩ restart::read_restart_states(), mo_init_states::variables_default_init(), and mo_mpr_restart::write_eff_params().

16.40.2.30 l1_maxinter

```
real(dp), dimension(:, :, :), allocatable, public mo_mpr_global_variables::l1_maxinter
```

Referenced by mo_mpr_startup::init_eff_params(), mo_mhm_eval::mhm_eval(), mo_mpr_eval::mpr_eval(), mo_↩ restart::read_restart_states(), mo_init_states::variables_default_init(), and mo_mpr_restart::write_eff_params().

16.40.2.31 l1_petlaicorfactor

```
real(dp), dimension(:, :, :), allocatable, public mo_mpr_global_variables::l1_petlaicorfactor
```

Referenced by mo_mpr_startup::init_eff_params(), mo_mhm_eval::mhm_eval(), mo_mpr_eval::mpr_eval(), mo_↩ restart::read_restart_states(), mo_init_states::variables_default_init(), and mo_mpr_restart::write_eff_params().

16.40.2.32 l1_prietayalpha

```
real(dp), dimension(:, :, :), allocatable, public mo_mpr_global_variables::l1_prietayalpha
```

Referenced by mo_mpr_startup::init_eff_params(), mo_mhm_eval::mhm_eval(), mo_mpr_eval::mpr_eval(), mo_↩ restart::read_restart_states(), mo_init_states::variables_default_init(), and mo_mpr_restart::write_eff_params().

16.40.2.33 l1_sealedthresh

```
real(dp), dimension(:, :, :), allocatable, public mo_mpr_global_variables::l1_sealedthresh
```

Referenced by mo_mpr_startup::init_eff_params(), mo_mhm_eval::mhm_eval(), mo_mpr_eval::mpr_eval(), mo_↩ restart::read_restart_states(), mo_init_states::variables_default_init(), and mo_mpr_restart::write_eff_params().

16.40.2.34 l1_soilmoistexp

```
real(dp), dimension(:, :, :), allocatable, public mo_mpr_global_variables::l1_soilmoistexp
```

Referenced by mo_mpr_startup::init_eff_params(), mo_mhm_eval::mhm_eval(), mo_mpr_eval::mpr_eval(), mo_↩ restart::read_restart_states(), mo_init_states::variables_default_init(), and mo_mpr_restart::write_eff_params().

16.40.2.35 l1_soilmoistfc

```
real(dp), dimension(:, :, :), allocatable, public mo_mpr_global_variables::l1_soilmoistfc
```

Referenced by mo_mpr_startup::init_eff_params(), mo_mhm_eval::mhm_eval(), mo_mpr_eval::mpr_eval(), mo_↩ restart::read_restart_states(), mo_init_states::variables_default_init(), and mo_mpr_restart::write_eff_params().

16.40.2.36 l1_soilmoistsat

```
real(dp), dimension(:, :, :), allocatable, public mo_mpr_global_variables::l1_soilmoistsat
```

Referenced by mo_mpr_startup::init_eff_params(), mo_mhm_eval::mhm_eval(), mo_mpr_eval::mpr_eval(), mo_↩ restart::read_restart_states(), mo_init_states::variables_default_init(), and mo_mpr_restart::write_eff_params().

16.40.2.37 l1_surfresist

```
real(dp), dimension(:, :, :), allocatable, public mo_mpr_global_variables::l1_surfresist
```

Referenced by mo_mpr_startup::init_eff_params(), mo_mhm_eval::mhm_eval(), mo_mpr_eval::mpr_eval(), mo_↩ restart::read_restart_states(), mo_init_states::variables_default_init(), and mo_mpr_restart::write_eff_params().

16.40.2.38 l1_tempthresh

```
real(dp), dimension(:, :, :), allocatable, public mo_mpr_global_variables::l1_tempthresh
```

Referenced by mo_mpr_startup::init_eff_params(), mo_mhm_eval::mhm_eval(), mo_mpr_eval::mpr_eval(), mo_↩ restart::read_restart_states(), mo_init_states::variables_default_init(), and mo_mpr_restart::write_eff_params().

16.40.2.39 l1_unsatthresh

```
real(dp), dimension(:, :, :), allocatable, public mo_mpr_global_variables::l1_unsatthresh
```

Referenced by mo_mpr_startup::init_eff_params(), mo_mhm_eval::mhm_eval(), mo_mpr_eval::mpr_eval(), mo_↩ restart::read_restart_states(), mo_init_states::variables_default_init(), and mo_mpr_restart::write_eff_params().

16.40.2.40 l1_wiltingpoint

```
real(dp), dimension(:, :, :), allocatable, public mo_mpr_global_variables::l1_wiltingpoint
```

Referenced by mo_mpr_startup::init_eff_params(), mo_mhm_eval::mhm_eval(), mo_mpr_eval::mpr_eval(), mo_↩ restart::read_restart_states(), mo_init_states::variables_default_init(), and mo_mpr_restart::write_eff_params().

16.40.2.41 lailut

```
real(dp), dimension(:, :), allocatable, public mo_mpr_global_variables::lailut
```

Referenced by mo_read_wrapper::read_data().

16.40.2.42 laiper

```
type(period), dimension(:), allocatable, public mo_mpr_global_variables::laiper
```

16.40.2.43 laiunitlist

```
integer(i4), dimension(:), allocatable, public mo_mpr_global_variables::laiunitlist
```

Referenced by mo_read_wrapper::read_data().

16.40.2.44 ngeounits

```
integer(i4), public mo_mpr_global_variables::ngeounits
```

Referenced by mo_mpr_read_config::mpr_read_config(), and mo_read_wrapper::read_data().

16.40.2.45 nlai

```
integer(i4), public mo_mpr_global_variables::nlai
```

Referenced by mo_mpr_startup::init_eff_params(), mo_prepare_gridded_lai::prepare_gridded_daily_lai_data(), mo_prepare_gridded_lai::prepare_gridded_mean_monthly_lai_data(), mo_read_wrapper::read_data(), mo_↩ restart::read_restart_states(), mo_mpr_restart::write_mpr_restart_files(), and mo_restart::write_restart_files().

16.40.2.46 nlaiclass

```
integer(i4), public mo_mpr_global_variables::nlaiclass
```

Referenced by mo_read_wrapper::read_data().

16.40.2.47 nsoilhorizons_mhm

```
integer(i4), public mo_mpr_global_variables::nsoilhorizons_mhm
```

Referenced by mo_soil_database::generate_soil_database(), mo_mpr_startup::init_eff_params(), mo_mpr_↩ startup::l0_check_input(), mo_mpr_startup::l0_variable_init(), mo_mhm_eval::mhm_eval(), mo_mhm_read_↩ config::mhm_read_config(), mo_multi_param_reg::mpr(), mo_mpr_read_config::mpr_read_config(), mo_write↩ _fluxes_states::newoutputdataset(), mo_read_wrapper::read_data(), mo_restart::read_restart_states(), mo_soil↩ _database::read_soil_lut(), mo_write_fluxes_states::updatedataset(), mo_init_states::variables_alloc(), mo_init_↩ states::variables_default_init(), mo_mpr_restart::write_mpr_restart_files(), and mo_restart::write_restart_files().

16.40.2.48 nsoiltypes

```
integer(i4), public mo_mpr_global_variables::nsoiltypes
```

Referenced by mo_soil_database::generate_soil_database(), mo_mpr_startup::l0_variable_init(), and mo_soil_↩ database::read_soil_lut().

16.40.2.49 soildb

```
type(soiltype), public mo_mpr_global_variables::soildb
```

Referenced by mo_soil_database::generate_soil_database(), mo_mpr_startup::l0_variable_init(), mo_multi_↩ param_reg::mpr(), mo_read_wrapper::read_data(), and mo_soil_database::read_soil_lut().

16.40.2.50 tillagedepth

```
real(dp), public mo_mpr_global_variables::tillagedepth
```

Referenced by mo_mpr_read_config::mpr_read_config(), and mo_soil_database::read_soil_lut().

16.40.2.51 timestep_lai_input

```
integer(i4), public mo_mpr_global_variables::timestep_lai_input
```

Referenced by mo_mpr_startup::l0_check_input(), mo_mhm_eval::mhm_eval(), mo_mpr_read_config::mpr_read↩ _config(), mo_prepare_gridded_lai::prepare_gridded_daily_lai_data(), and mo_read_wrapper::read_data().

16.41 mo_mpr_pet Module Reference

TODO: add description.

Functions/Subroutines

- subroutine, public pet_correctbylai (param, nodata, LCOVER0, LAI0, mask0, cell_id0, upp_row_L1, low_↩ row_L1, lef_col_L1, rig_col_L1, nL0_in_L1, L1_petLAIcorFactor)

estimate PET correction factor based on LAI at L1

- subroutine, public pet_correctbyasp (Id0, latitude_l0, Asp0, param, nodata, fAsp0)

  correction of PET

- subroutine, public priestley_taylor_alpha (LAI0, param, mask0, nodata, cell_id0, nL0_in_L1, Upp_row_L1, Low_row_L1, Lef_col_L1, Rig_col_L1, priestley_taylor_alpha1)

  Regionalization of priestley taylor alpha.

- subroutine, public bulksurface_resistance (LAI0, param, mask0, nodata, cell_id0, nL0_in_L1, Upp_row_L1, Low_row_L1, Lef_col_L1, Rig_col_L1, bulksurface_resistance1)

  Regionalization of bulk surface resistance.

16.41.1 Detailed Description

TODO: add description.

This module sets up pet correction factor at level-1 based on LAI

Authors

Mehmet Cuneyd Demirel, Simon Stisen

Date

May 2017

16.41.2 Function/Subroutine Documentation

16.41.2.1 bulksurface_resistance()

```
subroutine, public mo_mpr_pet::bulksurface_resistance (
             real(dp), dimension(:, :), intent(in) LAI0,
             real(dp), intent(in) param,
             logical, dimension(:, :), intent(in) mask0,
             real(dp), intent(in) nodata,
             integer(i4), dimension(:), intent(in) cell_id0,
             integer(i4), dimension(:), intent(in) nL0_in_L1,
             integer(i4), dimension(:), intent(in) Upp_row_L1,
             integer(i4), dimension(:), intent(in) Low_row_L1,
             integer(i4), dimension(:), intent(in) Lef_col_L1,
             integer(i4), dimension(:), intent(in) Rig_col_L1,
             real(dp), dimension(:, :), intent(out) bulksurface_resistance1 )
```

Regionalization of bulk surface resistance.

estimation of bulk surface resistance Global parameters needed (see mhm_parameter.nml):

- param(1) = stomatal_resistance

Parameters

in	*real(dp), dimension(:, :) :: LAI0*	LAI at level-0
in	*real(dp) :: param*	- global parameter
in	*logical, dimension(:, :) :: mask0*	mask at level 0
in	*real(dp) :: nodata*	- nodata value
in	*integer(i4), dimension(:) :: cell_id0*	Cell ids of hi res field

Parameters

in	*integer(i4), dimension(:) :: nL0_in_L1*	number of l0 cells within a l1 cell
in	*integer(i4), dimension(:) :: Upp_row_L1*	upper row of a l1 cell in l0 grid
in	*integer(i4), dimension(:) :: Low_row_L1*	lower row of a l1 cell in l0 grid
in	*integer(i4), dimension(:) :: Lef_col_L1*	left col of a l1 cell in l0 grid
in	*integer(i4), dimension(:) :: Rig_col_L1*	right col of a l1 cell in l0 grid
out	*real(dp), dimension(:, :) :: bulksurface_resistance1*	bulk surface resistance

Authors

Matthias Zink

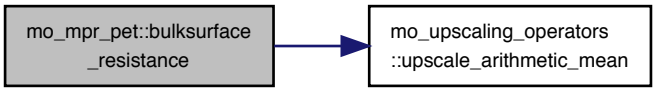
Date

Apr 2013

References mo_mpr_constants::lai_factor_surfresi, mo_mpr_constants::lai_offset_surfresi, mo_mpr_constants↩ ::max_surfresist, and mo_upscaling_operators::upscale_arithmetic_mean().

Referenced by mo_multi_param_reg::mpr().

Here is the call graph for this function:



Here is the caller graph for this function:

16.41.2.2 pet_correctbyasp()

```
subroutine, public mo_mpr_pet::pet_correctbyasp (
             integer(i4), dimension(:), intent(in) Id0,
             real(dp), dimension(:), intent(in) latitude_l0,
             real(dp), dimension(:), intent(in) Asp0,
             real(dp), dimension(3), intent(in) param,
             real(dp), intent(in) nodata,
             real(dp), dimension(:), intent(out) fAsp0 )
```

correction of PET

Correction of PET based on L0 aspect data. Global parameters needed (see mhm_parameter.nml):

- param(1) = minCorrectionFactorPET
- param(2) = maxCorrectionFactorPET
- param(3) = aspectTresholdPET

Parameters

in	*integer(i4), dimension(:) :: id0*	Level 0 cell id
in	*real(dp), dimension(:) :: latitude_l0*	latitude on l0
in	*real(dp), dimension(:) :: Asp0*	[degree] Aspect at Level 0
in	*real(dp), dimension(3) :: param*	process parameters
in	*real(dp) :: nodata*	- no data value
out	*real(dp), dimension(:) :: fAsp0*	PET correction for Aspect

Authors

Stephan Thober, Rohini Kumar

Date

Dec 2012

References mo_kind::i4.

Referenced by mo_multi_param_reg::mpr().

Here is the caller graph for this function:

16.41.2.3 pet_correctbylai()

```
subroutine, public mo_mpr_pet::pet_correctbylai (
             real(dp), dimension(5), intent(in) param,
             real(dp), intent(in) nodata,
             integer(i4), dimension(:), intent(in) LCOVER0,
             real(dp), dimension(:, :), intent(in) LAI0,
             logical, dimension(:, :), intent(in) mask0,
             integer(i4), dimension(:), intent(in) cell_id0,
             integer(i4), dimension(:), intent(in) upp_row_L1,
             integer(i4), dimension(:), intent(in) low_row_L1,
             integer(i4), dimension(:), intent(in) lef_col_L1,
             integer(i4), dimension(:), intent(in) rig_col_L1,
             integer(i4), dimension(:), intent(in) nL0_in_L1,
             real(dp), dimension(:, :), intent(inout) L1_petLAIcorFactor )
```

estimate PET correction factor based on LAI at L1

estimate PET correction factor based on LAI at L1 for a given Leaf Area Index field. Global parameters needed (see mhm_parameter.nml): Process Case 5:

- param(1) = PET_a_forest
- param(2) = PET_a_impervious
- param(3) = PET_a_pervious
- param(4) = PET_b
- param(5) = PET_c Example DSF=PET_a+PET_b∗(1-exp(PET_c∗LAI)) Similar to the crop coefficient concept Kc=a+b∗(1-exp(c∗LAI)) by Allen, R. G., L. S. Pereira, D. Raes, and M. Smith (1998), Crop evapotranspiration - Guidelines for computing crop water requirements., FAO Irrigation and drainage paper 56. See Chapter 9, Equation 97 http://www.fao.org/docrep/X0490E/x0490e0f.htm Date: 17/5/2017

Parameters

in	*real(dp), dimension(5) :: param*	parameters
in	*real(dp) :: nodata*	- nodata value
in	*integer(i4), dimension(:) :: LCOVER0*	Land cover at level 0
in	*real(dp), dimension(:, :) :: LAI0*	LAI at level-0
in	*logical, dimension(:, :) :: mask0*	mask at L0
in	*integer(i4), dimension(:) :: cell_id0*	Cell ids of hi res field
in	*integer(i4), dimension(:) :: upp_row_L1*	Upper row of hi res block
in	*integer(i4), dimension(:) :: low_row_L1*	Lower row of hi res block
in	*integer(i4), dimension(:) :: lef_col_L1*	Left column of hi res block
in	*integer(i4), dimension(:) :: rig_col_L1*	Right column of hi res block
in	*integer(i4), dimension(:) :: nL0_in_L1*	Number of L0 cells within a L1 cel
in,out	*real(dp), dimension(:, :) :: L1_petLAIcorFactor*	pet cor factor at level-1

Authors

M. Cuneyd Demirel and Simon Stisen from GEUS.dk

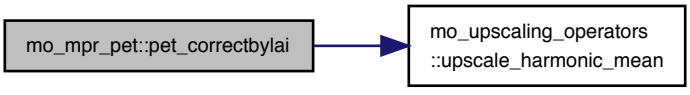
Date

May. 2017

References mo_upscaling_operators::upscale_harmonic_mean().

Referenced by mo_multi_param_reg::mpr().

Here is the call graph for this function:



Here is the caller graph for this function:

16.41.2.4 priestley_taylor_alpha()

```
subroutine, public mo_mpr_pet::priestley_taylor_alpha (
             real(dp), dimension(:, :), intent(in) LAI0,
             real(dp), dimension(:), intent(in) param,
             logical, dimension(:, :), intent(in) mask0,
             real(dp), intent(in) nodata,
             integer(i4), dimension(:), intent(in) cell_id0,
             integer(i4), dimension(:), intent(in) nL0_in_L1,
             integer(i4), dimension(:), intent(in) Upp_row_L1,
             integer(i4), dimension(:), intent(in) Low_row_L1,
             integer(i4), dimension(:), intent(in) Lef_col_L1,
             integer(i4), dimension(:), intent(in) Rig_col_L1,
             real(dp), dimension(:, :), intent(out) priestley_taylor_alpha1 )
```

Regionalization of priestley taylor alpha.

estimation of priestley taylor alpha Global parameters needed (see mhm_parameter.nml):

- param(1) = PriestleyTaylorCoeff
- param(2) = PriestleyTaylorLAIcorr

Parameters

in	*real(dp), dimension(:, :) :: LAI0*	LAI at level-0
in	*real(dp), dimension(:) :: param*	input parameter
in	*logical, dimension(:, :) :: mask0*	mask at level 0
in	*real(dp) :: nodata*	- nodata value
in	*integer(i4), dimension(:) :: cell_id0*	Cell ids of hi res field
in	*integer(i4), dimension(:) :: nL0_in_L1*	number of l0 cells within a l1 cell
in	*integer(i4), dimension(:) :: Upp_row_L1*	upper row of a l1 cell in l0 grid
in	*integer(i4), dimension(:) :: Low_row_L1*	lower row of a l1 cell in l0 grid
in	*integer(i4), dimension(:) :: Lef_col_L1*	left col of a l1 cell in l0 grid
in	*integer(i4), dimension(:) :: Rig_col_L1*	right col of a l1 cell in l0 grid
out	*real(dp), dimension(:, :) :: priestley_taylor_alpha1*	bulk surface resistance

Authors

Matthias Zink

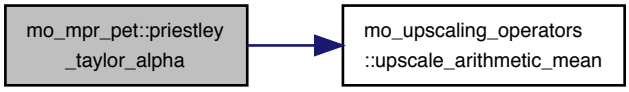
Date

Apr 2013

References mo_upscaling_operators::upscale_arithmetic_mean().

Referenced by mo_multi_param_reg::mpr().

Here is the call graph for this function:



Here is the caller graph for this function:

16.42 mo_mpr_read_config Module Reference

read mpr config

Functions/Subroutines

- subroutine, public mpr_read_config (file_namelist, unamelist, file_namelist_param, unamelist_param)
  Read the general config of mpr.

16.42.1 Detailed Description

read mpr config

This module contains all mpr subroutines related to reading the mpr configuration from file.

Authors

Stephan Thober

Date

Aug 2015

16.42.2 Function/Subroutine Documentation

16.42.2.1 mpr_read_config()

```
subroutine, public mo_mpr_read_config::mpr_read_config (
             character(*), intent(in) file_namelist,
             integer, intent(in) unamelist,
             character(*), intent(in) file_namelist_param,
             integer, intent(in) unamelist_param )
```

Read the general config of mpr.

Depending on the variable mrm_coupling_config, the mRM config is either read from mrm.nml and parameters from mrm_parameter.nml or copied from mHM.

Parameters

in	*character(∗) :: file_namelist*	
in	*integer :: unamelist*	
in	*character(∗) :: file_namelist_param*	
in	*integer :: unamelist_param*	

Authors

Stephan Thober

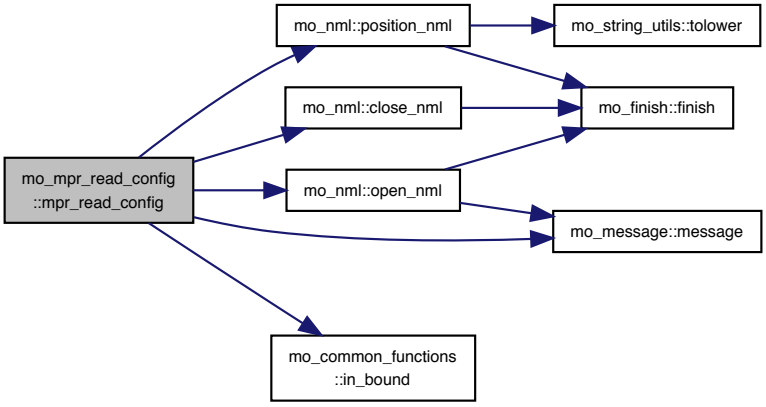
Date

Aug 2015

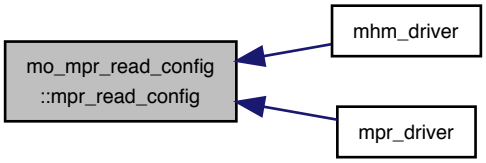
References mo_nml::close_nml(), mo_mpr_global_variables::dirgridded_lai, mo_common_constants::eps_dp, mo_mpr_global_variables::fracsealed_cityarea, mo_common_variables::global_parameters, mo_common_variables::global_parameters_name, mo_mpr_global_variables::horizondepth_mhm, mo_mpr_global_variables::iflag_soildb, mo_common_functions::in_bound(), mo_mpr_global_variables::inputformat_gridded_lai, mo_mpr_constants::maxgeounit, mo_common_constants::maxnobasins, mo_mpr_constants::maxnosoilhorizons, mo_message::message(), mo_common_variables::nbasins, mo_common_constants::ncolpars, mo_mpr_global_variables::ngeounits, mo_common_constants::nodata_dp, mo_mpr_global_variables::nsoilhorizons_mhm, mo_nml::open_nml(), mo_nml::position_nml(), mo_common_variables::processmatrix, mo_mpr_global_variables::tillagedepth, and mo_mpr_global_variables::timestep_lai_input.

Referenced by mhm_driver(), and mpr_driver().

Here is the call graph for this function:



Here is the caller graph for this function:



16.43 mo_mpr_restart Module Reference

reading and writing states, fluxes and configuration for restart of mHM.

Data Types

- interface unpack_field_and_write

  TODO: add description.

Functions/Subroutines

- subroutine, public write_mpr_restart_files (OutPath)

  write restart files for each basin
- subroutine, public write_eff_params (mask1, s1, e1, rows1, cols1, soil1, lcscenes, lais, nc)

  TODO: add description.
- subroutine unpack_field_and_write_1d_i4 (nc, var_name, var_dims, fill_value, data, mask, var_long_name)

- subroutine unpack_field_and_write_1d_dp (nc, var_name, var_dims, fill_value, data, mask, var_long_name)
- subroutine unpack_field_and_write_2d_dp (nc, var_name, var_dims, fill_value, data, mask, var_long_name)
- subroutine unpack_field_and_write_3d_dp (nc, var_name, var_dims, fill_value, data, mask, var_long_name)

16.43.1 Detailed Description

reading and writing states, fluxes and configuration for restart of mHM.

routines are seperated for reading and writing variables for:

- states and fluxes, and
- configuration. Reading of L11 configuration is also seperated from the rest, since it is only required when routing is activated.

Authors

Stephan Thober

Date

Jul 2013

16.43.2 Function/Subroutine Documentation

16.43.2.1 unpack_field_and_write_1d_dp()

```
subroutine mo_mpr_restart::unpack_field_and_write_1d_dp (
             type(ncdataset), intent(inout) nc,
             character(*), intent(in) var_name,
             type(ncdimension), dimension(:), intent(in) var_dims,
             real(dp), intent(in) fill_value,
             real(dp), dimension(:), intent(in) data,
             logical, dimension(:, :), intent(in) mask,
             character(*), intent(in), optional var_long_name )
```

References mo_kind::dp.

16.43.2.2 unpack_field_and_write_1d_i4()

```
subroutine mo_mpr_restart::unpack_field_and_write_1d_i4 (
             type(ncdataset), intent(inout) nc,
             character(*), intent(in) var_name,
             type(ncdimension), dimension(:), intent(in) var_dims,
             integer(i4), intent(in) fill_value,
             integer(i4), dimension(:), intent(in) data,
             logical, dimension(:, :), intent(in) mask,
             character(*), intent(in), optional var_long_name )
```

References mo_kind::i4.

16.43.2.3 unpack_field_and_write_2d_dp()

```
subroutine mo_mpr_restart::unpack_field_and_write_2d_dp (
             type(ncdataset), intent(inout) nc,
             character(*), intent(in) var_name,
             type(ncdimension), dimension(:), intent(in) var_dims,
             real(dp), intent(in) fill_value,
             real(dp), dimension(:, :), intent(in) data,
             logical, dimension(:, :), intent(in) mask,
             character(*), intent(in), optional var_long_name )
```

References mo_kind::dp, and mo_kind::i4.

16.43.2.4 unpack_field_and_write_3d_dp()

```
subroutine mo_mpr_restart::unpack_field_and_write_3d_dp (
             type(ncdataset), intent(inout) nc,
             character(*), intent(in) var_name,
             type(ncdimension), dimension(:), intent(in) var_dims,
             real(dp), intent(in) fill_value,
             real(dp), dimension(:, :, :), intent(in) data,
             logical, dimension(:, :), intent(in) mask,
             character(*), intent(in), optional var_long_name )
```

References mo_kind::dp, and mo_kind::i4.

16.43.2.5 write_eff_params()

```
subroutine, public mo_mpr_restart::write_eff_params (
             logical, dimension(:, :), intent(in), allocatable mask1,
             integer(i4), intent(in) s1,
             integer(i4), intent(in) e1,
             type(ncdimension), intent(in) rows1,
             type(ncdimension), intent(in) cols1,
             type(ncdimension), intent(in) soil1,
             type(ncdimension), intent(in) lcscenes,
             type(ncdimension), intent(in) lais,
             type(ncdataset), intent(inout) nc )
```

TODO: add description.

TODO: add description

Parameters

in	*logical, dimension(:, :) :: mask1*	mask at level 1
in	*integer(i4) :: s1*	start index at level 1
in	*integer(i4) :: e1*	end index at level 1
in	*type(NcDimension) :: rows1, cols1, soil1, lcscenes, lais*	
in	*type(NcDimension) :: rows1, cols1, soil1, lcscenes, lais*	
in	*type(NcDimension) :: rows1, cols1, soil1, lcscenes, lais*	
in	*type(NcDimension) :: rows1, cols1, soil1, lcscenes, lais*	
in	*type(NcDimension) :: rows1, cols1, soil1, lcscenes, lais*	
in,out	*type(NcDataset) :: nc*	

Authors

Robert Schweppe

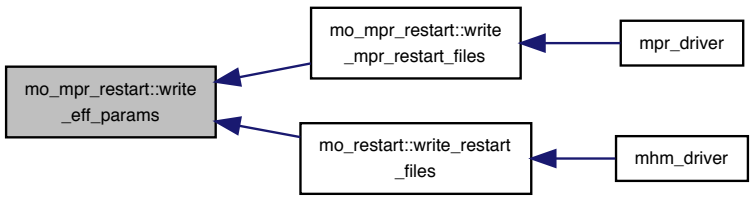
Date

Jun 2018

References mo_kind::i4, mo_mpr_global_variables::l1_aeroresist, mo_mpr_global_variables::l1_alpha, mo_mpr_global_variables::l1_degday, mo_mpr_global_variables::l1_degdayinc, mo_mpr_global_variables::l1_degdaymax, mo_mpr_global_variables::l1_degdaynopre, mo_mpr_global_variables::l1_fasp, mo_mpr_global_variables::l1_froots, mo_mpr_global_variables::l1_fsealed, mo_mpr_global_variables::l1_harsamcoeff, mo_mpr_global_variables::l1_jarvis_thresh_c1, mo_mpr_global_variables::l1_karstloss, mo_mpr_global_variables::l1_kbaseflow, mo_mpr_global_variables::l1_kfastflow, mo_mpr_global_variables::l1_kperco, mo_mpr_global_variables::l1_kslowflow, mo_mpr_global_variables::l1_maxinter, mo_mpr_global_variables::l1_petlaicorfactor, mo_mpr_global_variables::l1_prietayalpha, mo_mpr_global_variables::l1_sealedthresh, mo_mpr_global_variables::l1_soilmoistexp, mo_mpr_global_variables::l1_soilmoistfc, mo_mpr_global_variables::l1_soilmoistsat, mo_mpr_global_variables::l1_surfresist, mo_mpr_global_variables::l1_tempthresh, mo_mpr_global_variables::l1_unsatthresh, mo_mpr_global_variables::l1_wiltingpoint, mo_common_variables::lc_year_end, mo_common_variables::lc_year_start, mo_common_constants::nodata_dp, mo_common_constants::nodata_i4, and mo_common_variables::processmatrix.

Referenced by write_mpr_restart_files(), and mo_restart::write_restart_files().

Here is the caller graph for this function:



16.43.2.6 write_mpr_restart_files()

```
subroutine, public mo_mpr_restart::write_mpr_restart_files (
             character(256), dimension(:), intent(in) OutPath )
```

write restart files for each basin

write restart files for each basin. For each basin three restart files are written. These are xxx_states.nc, xxx_L11_config.nc, and xxx_config.nc (xxx being the three digit basin index). If a variable is added here, it should also be added in the read restart routines below. ADDITIONAL INFORMATION write_restart

Parameters

in	*character(256), dimension(:) :: OutPath*	Output Path for each basin

Authors

Stephan Thober

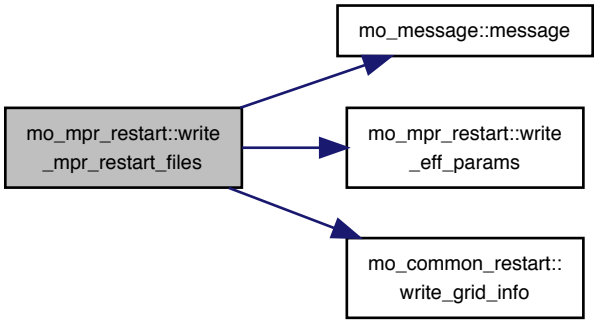
Date

Jun 2014

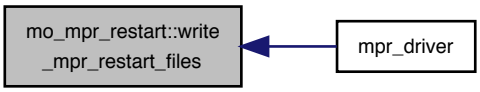
References mo_kind::i4, mo_common_variables::level1, mo_message::message(), mo_mpr_global_variables::nlai, mo_common_variables::nlcoverscene, mo_mpr_global_variables::nsoilhorizons_mhm, write_eff_params(), and mo_common_restart::write_grid_info().

Referenced by mpr_driver().

Here is the call graph for this function:



Here is the caller graph for this function:



16.44 mo_mpr_runoff Module Reference

multiscale parameter regionalization for runoff generation

Functions/Subroutines

- subroutine, public mpr_runoff (LCOVER0, mask0, SMs_FC0, slope_emp0, KsVar_H0, param, cell_id0, upp_row_L1, low_row_L1, lef_col_L1, rig_col_L1, nL0_in_L1, L1_HL1, L1_K0, L1_K1, L1_alpha)

  multiscale parameter regionalization for runoff parameters

16.44.1 Detailed Description

multiscale parameter regionalization for runoff generation

This contains the routine for multiscale parameter regionalization of the runoff parametrization.

Authors

Stephan Thober, Rohini Kumar

Date

Dec 2012

16.44.2 Function/Subroutine Documentation

16.44.2.1 mpr_runoff()

```
subroutine, public mo_mpr_runoff::mpr_runoff (
             integer(i4), dimension(:), intent(in) LCOVER0,
             logical, dimension(:, :), intent(in) mask0,
             real(dp), dimension(:), intent(in) SMs_FC0,
             real(dp), dimension(:), intent(in) slope_emp0,
             real(dp), dimension(:), intent(in) KsVar_H0,
             real(dp), dimension(5), intent(in) param,
             integer(i4), dimension(:), intent(in) cell_id0,
             integer(i4), dimension(:), intent(in) upp_row_L1,
             integer(i4), dimension(:), intent(in) low_row_L1,
             integer(i4), dimension(:), intent(in) lef_col_L1,
             integer(i4), dimension(:), intent(in) rig_col_L1,
             integer(i4), dimension(:), intent(in) nL0_in_L1,
             real(dp), dimension(:), intent(out) L1_HL1,
             real(dp), dimension(:), intent(out) L1_K0,
             real(dp), dimension(:), intent(out) L1_K1,
             real(dp), dimension(:), intent(out) L1_alpha )
```

multiscale parameter regionalization for runoff parameters

Perform the multiscale parameter regionalization for runoff global parameters (see mhm_parameter.nml). These are the following five parameters:

- param(1) = interflowStorageCapacityFactor
- param(2) = interflowRecession_slope
- param(3) = fastInterflowRecession_forest
- param(4) = slowInterflowRecession_Ks
- param(5) = exponentSlowInterflow

Parameters

in	*integer(i4), dimension(:) :: LCOVER0*	land cover at level 0
in	*logical, dimension(:, :) :: mask0*	mask at Level 0
in	*real(dp), dimension(:) :: SMs_FC0*	[-] soil mositure deficit from field
in	*real(dp), dimension(:) :: slope_emp0*	empirical quantile values F(slope)

Parameters

in	*real(dp), dimension(:) :: KsVar_H0*	[-] relative variability of saturated
in	*real(dp), dimension(5) :: param*	global parameters
in	*integer(i4), dimension(:) :: cell_id0*	Cell ids of hi res field
in	*integer(i4), dimension(:) :: upp_row_L1*	Upper row of hi res block
in	*integer(i4), dimension(:) :: low_row_L1*	Lower row of hi res block
in	*integer(i4), dimension(:) :: lef_col_L1*	Left column of hi res block
in	*integer(i4), dimension(:) :: rig_col_L1*	Right column of hi res block
in	*integer(i4), dimension(:) :: nL0_in_L1*	Number of L0 cells within a L1 cell
in	*real(dp) :: c2TSTu*	unit transformations
out	*real(dp), dimension(:) :: L1_HL1*	[10^-3 m] Threshhold water depth
out	*real(dp), dimension(:) :: L1_K0*	[10^-3 m] Recession coefficient
out	*real(dp), dimension(:) :: L1_K1*	[10^-3 m] Recession coefficient
out	*real(dp), dimension(:) :: L1_alpha*	[1] Exponent for the upper reservoir

Authors

Stephan Thober, Rohini Kumar

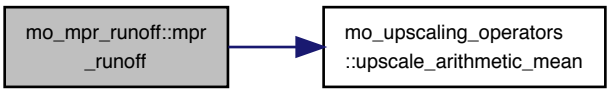
Date

Dec 2012

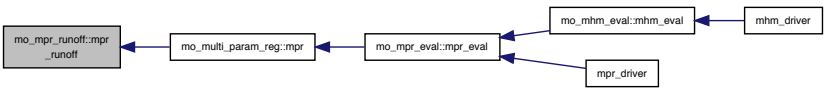
References mo_common_constants::nodata_dp, mo_common_constants::nodata_i4, and mo_upscaling_operators::upscale_arithmetic_mean().

Referenced by mo_multi_param_reg::mpr().

Here is the call graph for this function:



Here is the caller graph for this function:



16.45 mo_mpr_smhorizons Module Reference

setting up the soil moisture horizons

Functions/Subroutines

- subroutine, public mpr_smhorizons (param, processMatrix, iFlag_soil, nHorizons_mHM, HorizonDepth, L↩ COVER0, soilID0, nHorizons, nTillHorizons, thetaS_till, thetaFC_till, thetaPW_till, thetaS, thetaFC, thetaPW, Wd, Db, DbM, RZdepth, mask0, cell_id0, upp_row_L1, low_row_L1, lef_col_L1, rig_col_L1, nL0_in_L1, L1↩ _beta, L1_SMs, L1_FC, L1_PW, L1_fRoots)

  upscale soil moisture horizons

16.45.1 Detailed Description

setting up the soil moisture horizons

This module sets up the soil moisture horizons

Authors

Stephan Thober, Rohini Kumar

Date

Dec 2012

16.45.2 Function/Subroutine Documentation

16.45.2.1 mpr_smhorizons()

```
subroutine, public mo_mpr_smhorizons::mpr_smhorizons (
             real(dp), dimension(:), intent(in) param,
             integer(i4), dimension(:, :), intent(in) processMatrix,
             integer(i4), intent(in) iFlag_soil,
             integer(i4), intent(in) nHorizons_mHM,
             real(dp), dimension(:), intent(in) HorizonDepth,
             integer(i4), dimension(:), intent(in) LCOVER0,
             integer(i4), dimension(:, :), intent(in) soilID0,
             integer(i4), dimension(:), intent(in) nHorizons,
             integer(i4), dimension(:), intent(in) nTillHorizons,
             real(dp), dimension(:, :, :), intent(in) thetaS_till,
             real(dp), dimension(:, :, :), intent(in) thetaFC_till,
             real(dp), dimension(:, :, :), intent(in) thetaPW_till,
             real(dp), dimension(:, :), intent(in) thetaS,
             real(dp), dimension(:, :), intent(in) thetaFC,
             real(dp), dimension(:, :), intent(in) thetaPW,
             real(dp), dimension(:, :, :), intent(in) Wd,
             real(dp), dimension(:, :, :), intent(in) Db,
             real(dp), dimension(:, :), intent(in) DbM,
             real(dp), dimension(:), intent(in) RZdepth,
             logical, dimension(:, :), intent(in) mask0,
             integer(i4), dimension(:), intent(in) cell_id0,
             integer(i4), dimension(:), intent(in) upp_row_L1,
             integer(i4), dimension(:), intent(in) low_row_L1,
             integer(i4), dimension(:), intent(in) lef_col_L1,
             integer(i4), dimension(:), intent(in) rig_col_L1,
             integer(i4), dimension(:), intent(in) nL0_in_L1,
             real(dp), dimension(:, :), intent(inout) L1_beta,
```

```
real(dp), dimension(:, :), intent(inout) L1_SMs,
real(dp), dimension(:, :), intent(inout) L1_FC,
real(dp), dimension(:, :), intent(inout) L1_PW,
real(dp), dimension(:, :), intent(inout) L1_fRoots )
```

upscale soil moisture horizons

calculate soil properties at the level 1. Global parameters needed (see mhm_parameter.nml):

- param(1) = rootFractionCoefficient_forest
- param(2) = rootFractionCoefficient_impervious
- param(3) = rootFractionCoefficient_pervious
- param(4) = infiltrationShapeFactor

Parameters

in	*real(dp), dimension(:) :: param*	parameters
in	*integer(i4), dimension(:, :) :: processMatrix*	- matrix specifying user defined processes
in	*integer(i4) :: iFlag_soil*	- flags for handling multiple soil databases
in	*integer(i4) :: nHorizons_mHM*	- number of horizons to model
in	*real(dp), dimension(:) :: HorizonDepth*	[10^-3 m] horizon depth from surface, postive downwards
in	*integer(i4), dimension(:) :: LCOVER0*	Land cover at level 0
in	*integer(i4), dimension(:, :) :: soilID0*	soil ID at level 0
in	*integer(i4), dimension(:) :: nHorizons*	horizons per soil type
in	*integer(i4), dimension(:) :: nTillHorizons*	Number of Tillage horizons
in	*real(dp), dimension(:, :, :) :: thetaS_till*	saturated water content of soil horizons upto tillage depth, f(OM, management)
in	*real(dp), dimension(:, :, :) :: thetaFC_till*	Field capacity of tillage layers; LUC dependent, f(OM, management)
in	*real(dp), dimension(:, :, :) :: thetaPW_till*	Permament wilting point of tillage layers; LUC dependent, f(OM, management)
in	*real(dp), dimension(:, :) :: thetaS*	saturated water content of soil horizons after tillage depth
in	*real(dp), dimension(:, :) :: thetaFC*	Field capacity of deeper layers
in	*real(dp), dimension(:, :) :: thetaPW*	Permanent wilting point of deeper layers
in	*real(dp), dimension(:, :, :) :: Wd*	weights of mHM Horizons according to horizons provided in soil database
in	*real(dp), dimension(:, :, :) :: Db*	Bulk density
in	*real(dp), dimension(:, :) :: DbM*	mineral Bulk density
in	*real(dp), dimension(:) :: RZdepth*	[mm] Total soil depth
in	*logical, dimension(:, :) :: mask0*	mask at L0
in	*integer(i4), dimension(:) :: cell_id0*	Cell ids of hi res field
in	*integer(i4), dimension(:) :: upp_row_L1*	Upper row of hi res block
in	*integer(i4), dimension(:) :: low_row_L1*	Lower row of hi res block
in	*integer(i4), dimension(:) :: lef_col_L1*	Left column of hi res block
in	*integer(i4), dimension(:) :: rig_col_L1*	Right column of hi res block
in	*integer(i4), dimension(:) :: nL0_in_L1*	Number of L0 cells within a L1 cel
in,out	*real(dp), dimension(:, :) :: L1_beta*	Parameter that determines the relative contribution to SM, upscaled Bulk density
in,out	*real(dp), dimension(:, :) :: L1_SMs*	[10^-3 m] depth of saturated SM cont

Parameters

in,out	*real(dp), dimension(:, :) :: L1_FC*	[10^-3 m] field capacity
in,out	*real(dp), dimension(:, :) :: L1_PW*	[10^-3 m] permanent wilting point
in,out	*real(dp), dimension(:, :) :: L1_fRoots*	fraction of roots in soil horizons

Authors

Luis Samaniego, Rohini Kumar, Stephan Thober

Date

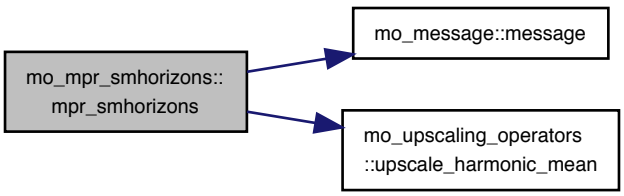
Dec 2012

in this case the second dimension of soilId0 = 1

References mo_message::message(), mo_common_constants::nodata_dp, and mo_upscaling_operators::upscale_harmonic_mean().

Referenced by mo_multi_param_reg::mpr().

Here is the call graph for this function:



Here is the caller graph for this function:

16.46 mo_mpr_soilmoist Module Reference

Multiscale parameter regionalization (MPR) for soil moisture.

Functions/Subroutines

- subroutine, public mpr_sm (param, is_present, nHorizons, nTillHorizons, sand, clay, DbM, ID0, soilId0, L↩ Cover0, thetaS_till, thetaFC_till, thetaPW_till, thetaS, thetaFC, thetaPW, Ks, Db, KsVar_H0, KsVar_V0, S↩ Ms_FC0)

multiscale parameter regionalization for soil moisture

- elemental pure subroutine pwp (Genu_Mual_n, Genu_Mual_alpha, thetaS, thetaPWP)

  Permanent Wilting point.
- elemental pure subroutine field_cap (thetaFC, Ks, thetaS, Genu_Mual_n)

  calculates the field capacity
- subroutine genuchten (thetaS, Genu_Mual_n, Genu_Mual_alpha, param, sand, clay, Db)

  calculates the Genuchten shape parameter
- subroutine hydro_cond (KS, param, sand, clay)

  calculates the hydraulic conductivity Ks

16.46.1 Detailed Description

Multiscale parameter regionalization (MPR) for soil moisture.

This module contains all routines required for parametrizing soil moisture processes.

Authors

Stephan Thober, Rohini Kumar

Date

Dec 2012

16.46.2 Function/Subroutine Documentation

16.46.2.1 field_cap()

```
elemental pure subroutine mo_mpr_soilmoist::field_cap (
             real(dp), intent(out) thetaFC,
             real(dp), intent(in) Ks,
             real(dp), intent(in) thetaS,
             real(dp), intent(in) Genu_Mual_n )
```

calculates the field capacity

estimate Field capacity; FC – Flux based approach (Twarakavi, et. al. 2009, WRR) According to the above reference FC is defined as the soil water content at which the drainage from a profile ceases under natural conditions. Since drainage from a soil profile in a simulation never becomes zero, we assume that drainage ceases when the bottom flux from the soil reaches a value that is equivalent to the minimum amount of precipitation that could be recorded (i.e. 0.01 cm/d == 1 mm/d). It is assumed that ThetaR = 0.0_dp ADDITIONAL INFORMATION Twarakavi, et. al. 2009, WRR

Parameters

out	*real(dp) :: thetaFC*	- Field capacity
in	*real(dp) :: Ks*	- saturated hydraulic conductivity
in	*real(dp) :: thetaS*	- saturated water content
in	*real(dp) :: Genu_Mual_n*	- Genuchten shape parameter

Authors

Stephan Thober, Rohini Kumar

Date

Dec 2012

References mo_mpr_constants::field_cap_c1, and mo_mpr_constants::field_cap_c2.

Referenced by mpr_sm().

Here is the caller graph for this function:

16.46.2.2 genuchten()

```
subroutine mo_mpr_soilmoist::genuchten (
            real(dp), intent(out) thetaS,
            real(dp), intent(out) Genu_Mual_n,
            real(dp), intent(out) Genu_Mual_alpha,
            real(dp), dimension(6), intent(in) param,
            real(dp), intent(in) sand,
            real(dp), intent(in) clay,
            real(dp), intent(in) Db )
```

calculates the Genuchten shape parameter

estimate SMs_till & van Genuchten's shape parameter (n) (Zacharias et al, 2007, soil Phy.) Global parameters needed (see mhm_parameter.nml):

- param(1) = PTF_lower66_5_constant
- param(2) = PTF_lower66_5_clay
- param(3) = PTF_lower66_5_Db
- param(4) = PTF_higher66_5_constant
- param(5) = PTF_higher66_5_clay
- param(6) = PTF_higher66_5_Db ADDITIONAL INFORMATION Zacharias et al, 2007, soil Phy.

Parameters

out	*real(dp) :: thetaS*	- saturated water content
out	*real(dp) :: Genu_Mual_n*	- van Genuchten shape parameter
out	*real(dp) :: Genu_Mual_alpha*	- van Genuchten shape parameter
in	*real(dp), dimension(6) :: param*	parameters
in	*real(dp) :: sand*	- [%] sand content
in	*real(dp) :: clay*	- [%] clay content
in	*real(dp) :: Db*	- [10^3 kg/m3] bulk density

Authors

Stephan Thober, Rohini Kumar

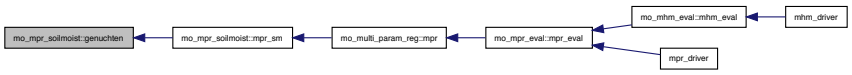
Date

Dec 2012

References mo_mpr_constants::vgenuchten_sandtresh, mo_mpr_constants::vgenuchtenn_c1, mo_mpr_↩ constants::vgenuchtenn_c10, mo_mpr_constants::vgenuchtenn_c11, mo_mpr_constants::vgenuchtenn_c12, mo↩ _mpr_constants::vgenuchtenn_c13, mo_mpr_constants::vgenuchtenn_c14, mo_mpr_constants::vgenuchtenn↩ _c15, mo_mpr_constants::vgenuchtenn_c16, mo_mpr_constants::vgenuchtenn_c17, mo_mpr_constants↩ ::vgenuchtenn_c18, mo_mpr_constants::vgenuchtenn_c2, mo_mpr_constants::vgenuchtenn_c3, mo_mpr_↩ constants::vgenuchtenn_c4, mo_mpr_constants::vgenuchtenn_c5, mo_mpr_constants::vgenuchtenn_c6, mo_↩ mpr_constants::vgenuchtenn_c7, mo_mpr_constants::vgenuchtenn_c8, and mo_mpr_constants::vgenuchtenn_c9.

Referenced by mpr_sm().

Here is the caller graph for this function:



16.46.2.3 hydro_cond()

```
subroutine mo_mpr_soilmoist::hydro_cond (
             real(dp), intent(out) KS,
             real(dp), dimension(4), intent(in) param,
             real(dp), intent(in) sand,
             real(dp), intent(in) clay )
```

calculates the hydraulic conductivity Ks

By default save this value of Ks, particularly for the deeper layers where OM content plays relatively low or no role Global parameters needed (see mhm_parameter.nml):

- param(1) = PTF_Ks_constant
- param(2) = PTF_Ks_sand
- param(3) = PTF_Ks_clay
- param(4) = PTF_Ks_curveSlope ADDITIONAL INFORMATION Written, Stephan Thober, Dec 2012

Parameters

out	*real(dp) :: KS*	
in	*real(dp), dimension(4) :: param*	
in	*real(dp) :: sand*	- [%] sand content
in	*real(dp) :: clay*	- [%] clay content

Authors

Stephan Thober, Rohini Kumar

Date

Dec 2012

References mo_mpr_constants::ks_c.

Referenced by mpr_sm().

Here is the caller graph for this function:

16.46.2.4 mpr_sm()

```
subroutine, public mo_mpr_soilmoist::mpr_sm (
             real(dp), dimension(13), intent(in) param,
             integer(i4), dimension(:), intent(in) is_present,
             integer(i4), dimension(:), intent(in) nHorizons,
             integer(i4), dimension(:), intent(in) nTillHorizons,
             real(dp), dimension(:, :), intent(in) sand,
             real(dp), dimension(:, :), intent(in) clay,
             real(dp), dimension(:, :), intent(in) DbM,
             integer(i4), dimension(:), intent(in) ID0,
             integer(i4), dimension(:, :), intent(in) soilId0,
             integer(i4), dimension(:), intent(in) LCover0,
             real(dp), dimension(:, :, :), intent(out) thetaS_till,
             real(dp), dimension(:, :, :), intent(out) thetaFC_till,
             real(dp), dimension(:, :, :), intent(out) thetaPW_till,
             real(dp), dimension(:, :), intent(out) thetaS,
             real(dp), dimension(:, :), intent(out) thetaFC,
             real(dp), dimension(:, :), intent(out) thetaPW,
             real(dp), dimension(:, :, :), intent(out) Ks,
             real(dp), dimension(:, :, :), intent(out) Db,
             real(dp), dimension(:), intent(out) KsVar_H0,
             real(dp), dimension(:), intent(out) KsVar_V0,
             real(dp), dimension(:), intent(out) SMs_FC0 )
```

multiscale parameter regionalization for soil moisture

This subroutine is a wrapper around all soil moisture parameter routines. This subroutine requires 13 parameters. These parameters have to correspond to the parameters in the original parameter array at the following locations: 10-12, 13-18, 27-30. Global parameters needed (see mhm_parameter.nml):

- param(1) = orgMatterContent_forest
- param(2) = orgMatterContent_impervious
- param(3) = orgMatterContent_pervious
- param(4) = PTF_lower66_5_constant

- param(5) = PTF_lower66_5_clay
- param(6) = PTF_lower66_5_Db
- param(7) = PTF_higher66_5_constant
- param(8) = PTF_higher66_5_clay
- param(9) = PTF_higher66_5_Db
- param(10) = PTF_Ks_constant
- param(11) = PTF_Ks_sand
- param(12) = PTF_Ks_clay
- param(13) = PTF_Ks_curveSlope

Parameters

in	*real(dp), dimension(13) :: param*	global parameters
in	*integer(i4), dimension(:) :: is_present*	indicates whether soiltype is present
in	*integer(i4), dimension(:) :: nHorizons*	Number of Horizons per soiltype
in	*integer(i4), dimension(:) :: nTillHorizons*	Number of Tillage Horizons
in	*real(dp), dimension(:, :) :: sand*	sand content
in	*real(dp), dimension(:, :) :: clay*	clay content
in	*real(dp), dimension(:, :) :: DbM*	mineral Bulk density
in	*integer(i4), dimension(:) :: ID0*	cell ids at level 0
in	*integer(i4), dimension(:, :) :: soilId0*	soil ids at level 0
in	*integer(i4), dimension(:) :: LCOVER0*	land cover ids at level 0
out	*real(dp), dimension(:, :, :) :: thetaS_till*	saturated soil moisture tillage layer
out	*real(dp), dimension(:, :, :) :: thetaFC_till*	field capacity tillage layer
out	*real(dp), dimension(:, :, :) :: thetaPW_till*	permanent wilting point tillage layer
out	*real(dp), dimension(:, :) :: thetaS*	saturated soil moisture
out	*real(dp), dimension(:, :) :: thetaFC*	field capacity
out	*real(dp), dimension(:, :) :: thetaPW*	permanent wilting point
out	*real(dp), dimension(:, :, :) :: Ks*	saturated hydraulic conductivity
out	*real(dp), dimension(:, :, :) :: Db*	Bulk density
out	*real(dp), dimension(:) :: KsVar_H0*	rel. var. of Ks for horizontal flow
out	*real(dp), dimension(:) :: KsVar_V0*	rel. var. of Ks for vertical flow
out	*real(dp), dimension(:) :: SMs_FC0*	soil mositure deficit from field cap. w.r.t to saturation

Authors

Stephan Thober, Rohini Kumar

Date

Dec 2012

here = ncells0

in this case the second dimension of soilId0 = 1

non-till

till layers

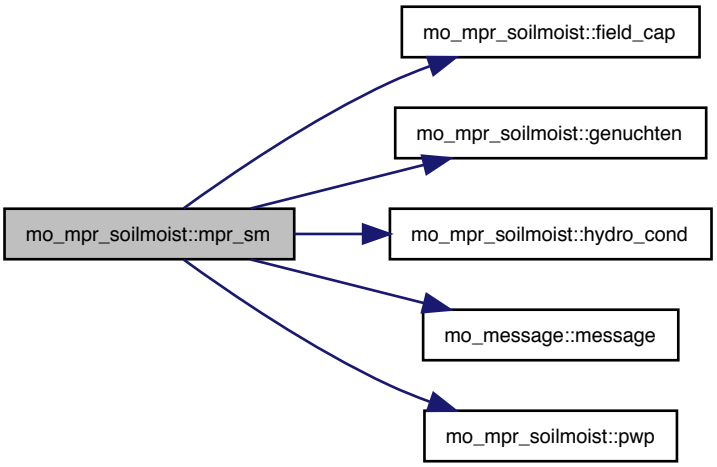
HORIZON

SOIL TYPE

References mo_mpr_constants::bulkdens_orgmatter, field_cap(), genuchten(), hydro_cond(), mo_mpr_global_↩ variables::iflag_soildb, mo_message::message(), mo_common_constants::nodata_dp, mo_common_constants↩ ::nodata_i4, and pwp().

Referenced by mo_multi_param_reg::mpr().

Here is the call graph for this function:



Here is the caller graph for this function:

16.46.2.5 pwp()

```
elemental pure subroutine mo_mpr_soilmoist::pwp (
             real(dp), intent(in) Genu_Mual_n,
             real(dp), intent(in) Genu_Mual_alpha,
             real(dp), intent(in) thetaS,
             real(dp), intent(out) thetaPWP )
```

Permanent Wilting point.

This subroutine calculates the permanent wilting point according to Zacharias et al. (2007, Soil Phy.) and using van Genuchten 1980's equation. For the water retention curve at a matrix potential of -1500 kPa, it is assumed that thetaR = 0. ADDITIONAL INFORMATION Zacharias et al. 2007, Soil Phy.

Parameters

in	*real(dp) :: Genu_Mual_n*	- Genuchten shape parameter
in	*real(dp) :: Genu_Mual_alpha*	- Genuchten shape parameter
in	*real(dp) :: thetaS*	- saturated water content
out	*real(dp) :: thetaPWP*	- Permanent Wilting point

Authors

Stephan Thober, Rohini Kumar

Date

Dec, 2012

References mo_mpr_constants::pwp_c, and mo_mpr_constants::pwp_matpot_thetar.

Referenced by mpr_sm().

Here is the caller graph for this function:

16.47 mo_mpr_startup Module Reference

Startup procedures for mHM.

Functions/Subroutines

- subroutine, public mpr_initialize

  Initialize main mHM variables.
- subroutine l0_check_input (iBasin)

  Check for errors in L0 input data.
- subroutine l0_variable_init (iBasin)

  level 0 variable initialization
- subroutine, public init_eff_params (ncells1)

  Allocation of space for mHM related L1 and L11 variables.

16.47.1 Detailed Description

Startup procedures for mHM.

This module initializes all variables required to run mHM. This module needs to be run only one time at the beginning of a simulation if re-starting files do not exist.

Authors

Luis Samaniego, Rohini Kumar

Date

Dec 2012

16.47.2 Function/Subroutine Documentation

16.47.2.1 init_eff_params()

```
subroutine, public mo_mpr_startup::init_eff_params (
            integer(i4), intent(in) ncells1 )
```

Allocation of space for mHM related L1 and L11 variables.

Allocation of space for mHM related L1 and L11 variables (e.g., states, fluxes, and parameters) for a given basin. Variables allocated here is defined in them mo_global_variables.f90 file. After allocating any variable in this routine, initalize them in the following variables_default_init subroutine:

Parameters

in	*integer(i4) :: ncells1*	

Authors

Rohini Kumar

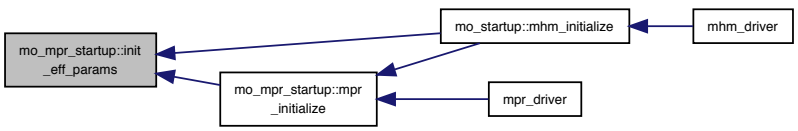
Date

Jan 2013

References mo_mpr_global_variables::l1_aeroresist, mo_mpr_global_variables::l1_alpha, mo_mpr_global_variables::l1_degday, mo_mpr_global_variables::l1_degdayinc, mo_mpr_global_variables::l1_degdaymax, mo_mpr_global_variables::l1_degdaynopre, mo_mpr_global_variables::l1_fasp, mo_mpr_global_variables::l1_froots, mo_mpr_global_variables::l1_fsealed, mo_mpr_global_variables::l1_harsamcoeff, mo_mpr_global_variables::l1_jarvis_thresh_c1, mo_mpr_global_variables::l1_karstloss, mo_mpr_global_variables::l1_kbaseflow, mo_mpr_global_variables::l1_kfastflow, mo_mpr_global_variables::l1_kperco, mo_mpr_global_variables::l1_kslowflow, mo_mpr_global_variables::l1_maxinter, mo_mpr_global_variables::l1_petlaicorfactor, mo_mpr_global_variables::l1_prietayalpha, mo_mpr_global_variables::l1_sealedthresh, mo_mpr_global_variables::l1_soilmoistexp, mo_mpr_global_variables::l1_soilmoistfc, mo_mpr_global_variables::l1_soilmoistsat, mo_mpr_global_variables::l1_surfresist, mo_mpr_global_variables::l1_tempthresh, mo_mpr_global_variables::l1_unsatthresh, mo_mpr_global_variables::l1_wiltingpoint, mo_mpr_global_variables::nlai, mo_common_variables::nlcoverscene, mo_mpr_global_variables::nsoilhorizons_mhm, mo_common_constants::p1_initstatefluxes, and mo_common_constants::yearmonths_i4.

Referenced by mo_startup::mhm_initialize(), and mpr_initialize().

Here is the caller graph for this function:



16.47.2.2 l0_check_input()

```
subroutine mo_mpr_startup::l0_check_input (
            integer(i4), intent(in) iBasin )
```

Check for errors in L0 input data.

Check for possible errors in input data (morphological and land cover) at level-0

Parameters

in	*integer(i4) :: iBasin*	basin id

Authors

Rohini Kumar

Date

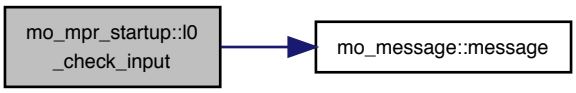
Jan 2013

by default; when iFlag_soilDB = 0

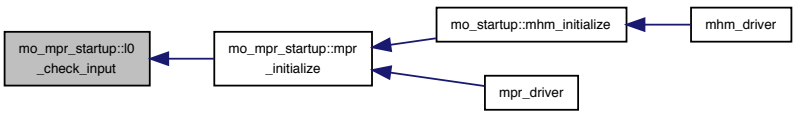
References mo_common_constants::eps_dp, mo_mpr_global_variables::iflag_soildb, mo_mpr_global_variables::l0_asp, mo_common_variables::l0_elev, mo_mpr_global_variables::l0_geounit, mo_mpr_global_variables::l0_gridded_lai, mo_common_variables::l0_lcover, mo_mpr_global_variables::l0_slope, mo_mpr_global_variables::l0_soilid, mo_common_variables::level0, mo_message::message(), mo_message::message_text, mo_common_variables::nlcoverscene, mo_mpr_global_variables::nsoilhorizons_mhm, and mo_mpr_global_variables::timestep_lai_input.

Referenced by mpr_initialize().

Here is the call graph for this function:



Here is the caller graph for this function:



16.47.2.3 l0_variable_init()

```
subroutine mo_mpr_startup::l0_variable_init (
             integer(i4), intent(in) iBasin )
```

level 0 variable initialization

following tasks are performed for L0 data sets

- cell id & numbering
- storage of cell cordinates (row and coloum id)
- empirical dist. of terrain slope
- flag to determine the presence of a particular soil id in this configuration of the model run If a variable is added or removed here, then it also has to be added or removed in the subroutine config_variables_set in module mo_restart and in the subroutine set_config in module mo_set_netcdf_restart

Parameters

in	*integer(i4) :: iBasin*	basin id

Authors

Rohini Kumar

Date

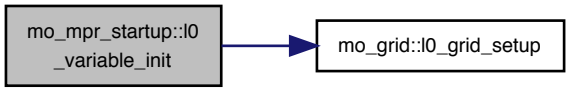
Jan 2013

by default; when iFlag_soilDB = 0

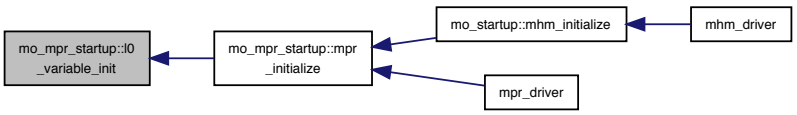
References mo_mpr_global_variables::iflag_soildb, mo_grid::l0_grid_setup(), mo_mpr_global_variables::l0_slope, mo_mpr_global_variables::l0_slope_emp, mo_mpr_global_variables::l0_soilid, mo_common_variables::level0, mo_mpr_global_variables::nsoilhorizons_mhm, mo_mpr_global_variables::nsoiltypes, and mo_mpr_global_↩ variables::soildb.

Referenced by mpr_initialize().

Here is the call graph for this function:



Here is the caller graph for this function:



16.47.2.4 mpr_initialize()

```
subroutine, public mo_mpr_startup::mpr_initialize ( )
```

Initialize main mHM variables.

Initialize main mHM variables for a given basin. Calls the following procedures in this order:

- Constant initialization.
- Generate soil database.
- Checking inconsistencies input fields.
- Variable initialization at level-0.
- Variable initialization at level-1.
- Variable initialization at level-11.
- Space allocation of remaining variable/parameters. Global variables will be used at this stage.

Authors

Luis Samaniego, Rohini Kumar

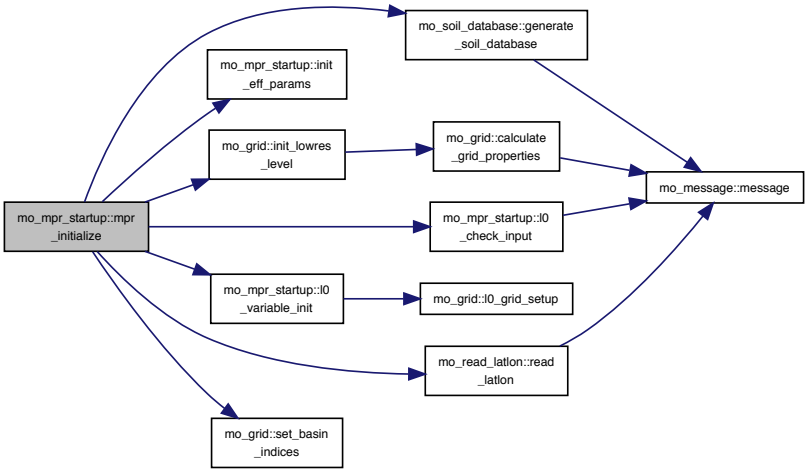
Date

Dec 2012

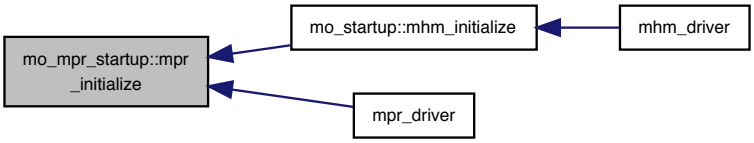
References mo_soil_database::generate_soil_database(), mo_kind::i4, init_eff_params(), mo_grid::init_lowres_↩ level(), mo_common_variables::l0_basin, l0_check_input(), mo_common_variables::l0_l1_remap, l0_variable_init(), mo_common_variables::level0, mo_common_variables::level1, mo_common_variables::nbasins, mo_read_latlon↩ ::read_latlon(), mo_common_variables::resolutionhydrology, and mo_grid::set_basin_indices().

Referenced by mo_startup::mhm_initialize(), and mpr_driver().

Here is the call graph for this function:



Here is the caller graph for this function:



16.48 mo_mrm_constants Module Reference

Provides mRM specific constants.

Variables

- integer(i4), parameter, public noutflxstate = 1_i4
- integer(i4), parameter, public nroutingstates = 2
- integer(i4), parameter, public maxnogauges = 50_i4

- real(dp), parameter, public rout_space_weight = 0._dp
- real(dp), parameter, public deltah = 5.000_dp
- real(dp), dimension(19), parameter given_ts = (/ 60._dp, 120._dp, 180._dp, 240._dp, 300._dp, 360._dp, 600._dp, 720._dp, 900._dp, 1200._dp, 1800._dp, 3600._dp, 7200._dp, 10800._dp, 14400._dp, 21600._dp, 28800._dp, 43200._dp, 86400._dp/)

16.48.1 Detailed Description

Provides mRM specific constants.

Provides mRM specific constants such as flood plain elevation.

Authors

Stephan Thober

Date

Aug 2015

16.48.2 Variable Documentation

16.48.2.1 deltah

```
real(dp), parameter, public mo_mrm_constants::deltah = 5.000_dp
```

Referenced by mo_mrm_net_startup::moveup().

16.48.2.2 given_ts

```
real(dp), dimension(19), parameter mo_mrm_constants::given_ts = (/ 60._dp, 120._dp, 180._dp,
240._dp, 300._dp, 360._dp, 600._dp, 720._dp, 900._dp, 1200._dp, 1800._dp, 3600._dp, 7200._dp,
10800._dp, 14400._dp, 21600._dp, 28800._dp, 43200._dp, 86400._dp/)
```

Referenced by mo_mrm_mpr::mrm_init_param(), and mo_mrm_mpr::mrm_update_param().

16.48.2.3 maxnogauges

```
integer(i4), parameter, public mo_mrm_constants::maxnogauges = 50_i4
```

Referenced by mo_mrm_read_config::mrm_read_config().

16.48.2.4 noutflxstate

```
integer(i4), parameter, public mo_mrm_constants::noutflxstate = 1_i4
```

16.48.2.5 nroutingstates

```
integer(i4), parameter, public mo_mrm_constants::nroutingstates = 2
```

Referenced by mo_mrm_restart::mrm_read_restart_states(), mo_mrm_restart::mrm_write_restart(), and mo_↩ mrm_init::variables_alloc_routing().

16.48.2.6 rout_space_weight

```
real(dp), parameter, public mo_mrm_constants::rout_space_weight = 0._dp
```

Referenced by mo_mrm_mpr::mrm_update_param().

16.49 mo_mrm_eval Module Reference

Runs mrm with a specific parameter set and returns required variables, e.g. runoff.

Functions/Subroutines

- subroutine, public mrm_eval (parameterset, runoff, sm_opti, basin_avg_tws, neutrons_opti, et_opti)
  Runs mrm with a specific parameter set and returns required variables, e.g. runoff.

16.49.1 Detailed Description

Runs mrm with a specific parameter set and returns required variables, e.g. runoff.

Runs mrm with a specific parameter set and returns required variables, e.g. runoff.

Authors

Stephan Thober

Date

Sep 2015

16.49.2 Function/Subroutine Documentation

16.49.2.1 mrm_eval()

```
subroutine, public mo_mrm_eval::mrm_eval (
             real(dp), dimension(:), intent(in) parameterset,
             real(dp), dimension(:, :), intent(out), optional, allocatable runoff,
             real(dp), dimension(:, :), intent(out), optional, allocatable sm_opti,
             real(dp), dimension(:, :), intent(out), optional, allocatable basin_avg_tws,
             real(dp), dimension(:, :), intent(out), optional, allocatable neutrons_opti,
             real(dp), dimension(:, :), intent(out), optional, allocatable et_opti )
```

Runs mrm with a specific parameter set and returns required variables, e.g. runoff.

Runs mrm with a specific parameter set and returns required variables, e.g. runoff.

Parameters

in	*real(dp), dimension(:) :: parameterset*	a set of global parameter (gamma) to run mHM, DIMENSION [no. of global_Parameters]
out	*real(dp), dimension(:, :), optional :: runoff*	returns runoff time series, DIMENSION [nTimeSteps, nGaugesTotal]
out	*real(dp), dimension(:, :), optional :: sm_opti*	dim1=ncells, dim2=time
out	*real(dp), dimension(:, :), optional :: basin_avg_tws*	dim1=time dim2=nBasins
out	*real(dp), dimension(:, :), optional :: neutrons_opti*	dim1=ncells, dim2=time
out	*real(dp), dimension(:, :), optional :: et_opti*	dim1=ncells, dim2=time

Authors

Stephan Thober

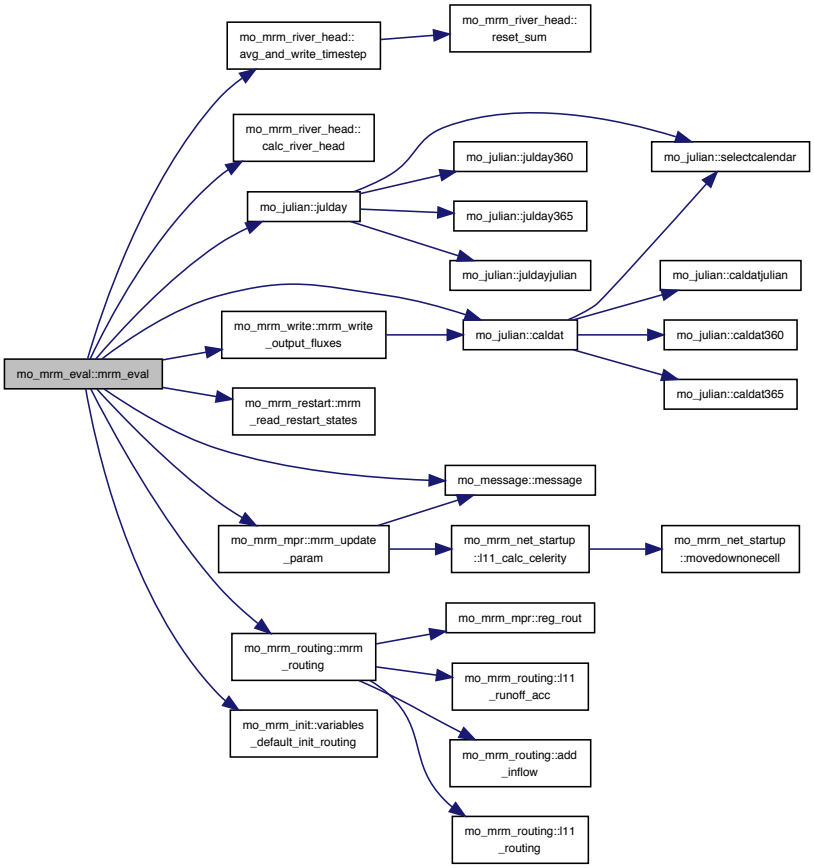
Date

Sep 2015

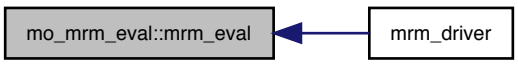
References mo_mrm_river_head::avg_and_write_timestep(), mo_mrm_global_variables::basin_mrm, mo_mrm_river_head::calc_river_head(), mo_julian::caldat(), mo_common_mhm_mrm_variables::dirrestartin, mo_kind::dp, mo_mrm_global_variables::gw_coupling, mo_common_constants::hoursecs, mo_kind::i4, mo_mrm_global_variables::inflowgauge, mo_julian::julday(), mo_mrm_global_variables::l0_river_head_mon_sum, mo_mrm_global_variables::l11_c1, mo_mrm_global_variables::l11_c2, mo_mrm_global_variables::l11_fromn, mo_mrm_global_variables::l11_l1_id, mo_mrm_global_variables::l11_length, mo_mrm_global_variables::l11_netperm, mo_mrm_global_variables::l11_nlinkfracfpimp, mo_mrm_global_variables::l11_noutlets, mo_mrm_global_variables::l11_qmod, mo_mrm_global_variables::l11_qout, mo_mrm_global_variables::l11_qtin, mo_mrm_global_variables::l11_qtr, mo_mrm_global_variables::l11_slope, mo_mrm_global_variables::l11_ton, mo_mrm_global_variables::l11_tsrout, mo_mrm_global_variables::l1_l11_id, mo_mrm_global_variables::l1_total_runoff_in, mo_common_mhm_mrm_variables::lcyearid, mo_common_variables::level1, mo_mrm_global_variables::level11, mo_message::message(), mo_mrm_restart::mrm_read_restart_states(), mo_mrm_routing::mrm_routing(), mo_mrm_global_variables::mrm_runoff, mo_mrm_mpr::mrm_update_param(), mo_mrm_write::mrm_write_output_fluxes(), mo_common_variables::nbasins, mo_common_mhm_mrm_variables::ntstepday, mo_common_mhm_mrm_variables::optimize, mo_mrm_global_variables::outputflxstate_mrm, mo_common_variables::processmatrix, mo_common_mhm_mrm_variables::read_restart, mo_common_variables::resolutionhydrology, mo_common_mhm_mrm_variables::resolutionrouting, mo_common_mhm_mrm_variables::simper, mo_common_mhm_mrm_variables::timestep, mo_mrm_global_variables::timestep_model_outputs_mrm, mo_mrm_init::variables_default_init_routing(), and mo_common_mhm_mrm_variables::warmingdays.

Referenced by mrm_driver().

Here is the call graph for this function:



Here is the caller graph for this function:



16.50 mo_mrm_file Module Reference

Provides file names and units for mRM.

Variables

- character(len=∗), parameter version = '1.0'

  Current mHM model version.

- character(len=∗), parameter version_date = 'May 2019'

  Time of current mHM model version release.
- character(len=∗), parameter file_main = 'mrm_driver.f90'

  Driver file.
- character(len=∗), parameter file_namelist_mrm = 'mrm.nml'

  Namelist file name.
- integer, parameter unamelist_mrm = 40

  Unit for namelist.
- character(len=∗), parameter file_namelist_param_mrm = 'mrm_parameter.nml'

  Parameter namelists file name.
- integer, parameter unamelist_param_mrm = 41

  Unit for namelist.
- character(len=∗), parameter file_facc = 'facc.asc'
- integer, parameter ufacc = 56

  Unit for flow accumulation input data file.
- character(len=∗), parameter file_fdir = 'fdir.asc'

  flow direction input data file
- integer, parameter ufdir = 57

  Unit for flow direction input data file.
- character(len= ∗), parameter file_slope = 'slope.asc'

  flow direction input data file
- integer, parameter uslope = 59

  Unit for flow direction input data file.
- character(len=∗), parameter file_gaugeloc = 'idgauges.asc'

  gauge location input data file
- integer, parameter ugaugeloc = 62

  Unit for gauge location input data file.
- integer, parameter udischarge = 66

  unit for discharge time series
- character(len=∗), parameter file_defoutput = 'mrm_outputs.nml'

  file defining mRM's outputs
- integer, parameter udefoutput = 67

  Unit for file defining mRM's outputs.
- character(len=∗), parameter file_config = 'ConfigFile.log'

  file defining mHM's outputs
- integer, parameter uconfig = 68

  Unit for file defining mHM's outputs.
- character(len=∗), parameter file_daily_discharge = 'daily_discharge.out'

  file defining optimazation outputs
- integer, parameter udaily_discharge = 74

  Unit for file optimazation outputs.
- character(len=∗), parameter ncfile_discharge = 'discharge.nc'

  file defining optimazation outputs
- character(len=∗), parameter file_mrm_output = 'mRM_Fluxes_States.nc'

  file containing mrm output
- character(len=∗), parameter file_gw_output = 'mRM_gw_Fluxes_States.nc'

  file containing mrm output for groundwater coupling

16.50.1 Detailed Description

Provides file names and units for mRM.

Provides all filenames as well as all units used for the multiscale Routing Model mRM.

Authors

Matthias Cuntz, Stephan Thober

Date

Aug 2015

16.50.2 Variable Documentation

16.50.2.1 file_config

```
character(len = *), parameter mo_mrm_file::file_config = 'ConfigFile.log'
```

file defining mHM's outputs

16.50.2.2 file_daily_discharge

```
character(len = *), parameter mo_mrm_file::file_daily_discharge = 'daily_discharge.out'
```

file defining optimazation outputs

Referenced by mo_mrm_write::write_daily_obs_sim_discharge().

16.50.2.3 file_defoutput

```
character(len = *), parameter mo_mrm_file::file_defoutput = 'mrm_outputs.nml'
```

file defining mRM's outputs

Referenced by mo_mrm_init::config_output(), mo_mrm_read_config::mrm_read_config(), and mo_mrm_init::print↩_startup_message().

16.50.2.4 file_facc

```
character(len = *), parameter mo_mrm_file::file_facc = 'facc.asc'
```

Referenced by mo_mrm_read_data::mrm_read_l0_data().

16.50.2.5 file_fdir

```
character(len = *), parameter mo_mrm_file::file_fdir = 'fdir.asc'
```

flow direction input data file

Referenced by mo_mrm_read_data::mrm_read_l0_data().

16.50.2.6 file_gaugeloc

```
character(len = *), parameter mo_mrm_file::file_gaugeloc = 'idgauges.asc'
```

gauge location input data file

Referenced by mo_mrm_read_data::mrm_read_l0_data().

16.50.2.7 file_gw_output

```
character(len = *), parameter mo_mrm_file::file_gw_output = 'mRM_gw_Fluxes_States.nc'
```

file containing mrm output for groundwater coupling

16.50.2.8 file_main

```
character(len = *), parameter mo_mrm_file::file_main = 'mrm_driver.f90'
```

Driver file.

Referenced by mo_mrm_init::print_startup_message().

16.50.2.9 file_mrm_output

```
character(len = *), parameter mo_mrm_file::file_mrm_output = 'mRM_Fluxes_States.nc'
```

file containing mrm output

Referenced by mo_mrm_write_fluxes_states::createoutputfile().

16.50.2.10 file_namelist_mrm

```
character(len = *), parameter mo_mrm_file::file_namelist_mrm = 'mrm.nml'
```

Namelist file name.

Referenced by mo_mrm_init::config_output(), and mrm_driver().

16.50.2.11 file_namelist_param_mrm

```
character(len = *), parameter mo_mrm_file::file_namelist_param_mrm = 'mrm_parameter.nml'
```

Parameter namelists file name.

Referenced by mo_mrm_init::config_output(), and mrm_driver().

16.50.2.12 file_slope

```
character(len=*), parameter mo_mrm_file::file_slope = 'slope.asc'
```

flow direction input data file

16.50.2.13 ncfile_discharge

```
character(len = *), parameter mo_mrm_file::ncfile_discharge = 'discharge.nc'
```

file defining optimazation outputs

Referenced by mo_mrm_write::write_daily_obs_sim_discharge().

16.50.2.14 uconfig

```
integer, parameter mo_mrm_file::uconfig = 68
```

Unit for file defining mHM's outputs.

16.50.2.15 udaily_discharge

```
integer, parameter mo_mrm_file::udaily_discharge = 74
```

Unit for file optimazation outputs.

Referenced by mo_mrm_write::write_daily_obs_sim_discharge().

16.50.2.16 udefoutput

```
integer, parameter mo_mrm_file::udefoutput = 67
```

Unit for file defining mRM's outputs.

Referenced by mo_mrm_read_config::mrm_read_config().

16.50.2.17 udischarge

```
integer, parameter mo_mrm_file::udischarge = 66
```

unit for discharge time series

Referenced by mo_mrm_read_data::mrm_read_discharge().

16.50.2.18 ufacc

```
integer, parameter mo_mrm_file::ufacc = 56
```

Unit for flow accumulation input data file.

Referenced by mo_mrm_read_data::mrm_read_l0_data().

16.50.2.19 ufdir

```
integer, parameter mo_mrm_file::ufdir = 57
```

Unit for flow direction input data file.

Referenced by mo_mrm_read_data::mrm_read_l0_data().

16.50.2.20 ugaugeloc

```
integer, parameter mo_mrm_file::ugaugeloc = 62
```

Unit for gauge location input data file.

Referenced by mo_mrm_read_data::mrm_read_l0_data().

16.50.2.21 unamelist_mrm

```
integer, parameter mo_mrm_file::unamelist_mrm = 40
```

Unit for namelist.

Referenced by mrm_driver().

16.50.2.22 unamelist_param_mrm

```
integer, parameter mo_mrm_file::unamelist_param_mrm = 41
```

Unit for namelist.

Referenced by mrm_driver().

16.50.2.23 uslope

```
integer, parameter mo_mrm_file::uslope = 59
```

Unit for flow direction input data file.

16.50.2.24 version

```
character(len = *), parameter mo_mrm_file::version = '1.0'
```

Current mHM model version.

Referenced by mo_mrm_write_fluxes_states::createoutputfile(), mo_mrm_init::print_startup_message(), and mo↩_mrm_write::write_configfile().

16.50.2.25 version_date

```
character(len = *), parameter mo_mrm_file::version_date = 'May 2019'
```

Time of current mHM model version release.

Referenced by mo_mrm_init::print_startup_message().

16.51 mo_mrm_global_variables Module Reference

Global variables for mRM only.

Data Types

- type basininfo_mrm
- type gaugingstation

Variables

- logical is_start
- integer(i4) timestep_model_outputs_mrm
- logical, dimension(noutflxstate) outputflxstate_mrm
- character(256), dimension(:), allocatable, public dirgauges
- character(256), dimension(:), allocatable, public dirtotalrunoff
- character(256), public filenametotalrunoff
- character(256), public varnametotalrunoff
- character(256), dimension(:), allocatable, public dirbankfullrunoff
- integer(i4), public ntstepday
- type(grid), dimension(:), allocatable, target, public level11
- type(gridremapper), dimension(:), allocatable, public l0_l11_remap
- type(gridremapper), dimension(:), allocatable, public l1_l11_remap
- real(dp), dimension(:, :), allocatable, public mrm_runoff
- integer(i4), public ngaugestotal
- integer(i4), public ninflowgaugestotal
- integer(i4), public nmeasperday
- type(gaugingstation), public gauge
- type(gaugingstation), public inflowgauge
- type(basininfo_mrm), dimension(:), allocatable, target, public basin_mrm
- integer(i4), dimension(:), allocatable, public l0_gaugeloc
- integer(i4), dimension(:), allocatable, public l0_inflowgaugeloc
- integer(i4), dimension(:), allocatable, public l0_facc
- integer(i4), dimension(:), allocatable, public l0_fdir
- integer(i4), dimension(:), allocatable, public l0_drasc
- integer(i4), dimension(:), allocatable, public l0_dracell
- integer(i4), dimension(:), allocatable, public l0_streamnet
- integer(i4), dimension(:), allocatable, public l0_floodplain
- integer(i4), dimension(:), allocatable, public l0_noutlet
- real(dp), dimension(:), allocatable, public l0_celerity
- integer(i4), dimension(:), allocatable, public l11_l1_id
- real(dp), dimension(:, :), allocatable, public l1_total_runoff_in
- integer(i4), dimension(:,:), allocatable, public l11_cellcoor
- integer(i4), dimension(:), allocatable, public l1_l11_id
- real(dp), dimension(:), allocatable, public l11_areacell

- real(dp), dimension(:), allocatable, public l11_facc
- integer(i4), dimension(:), allocatable, public l11_fdir
- integer(i4), dimension(:), allocatable, public l11_noutlets
- real(dp), dimension(:), allocatable, public l11_celerity
- real(dp), dimension(:), allocatable, public l11_meandering
- real(dp), dimension(:), allocatable, public l11_linkin_facc
- integer(i4), dimension(:), allocatable, public l11_rowout
- integer(i4), dimension(:), allocatable, public l11_colout
- real(dp), dimension(:), allocatable, public l11_qmod
- real(dp), dimension(:), allocatable, public l11_qout
- real(dp), dimension(:, :), allocatable, public l11_qtin
- real(dp), dimension(:, :), allocatable, public l11_qtr
- integer(i4), dimension(:), allocatable, public l11_fromn
- integer(i4), dimension(:), allocatable, public l11_ton
- integer(i4), dimension(:), allocatable, public l11_netperm
- integer(i4), dimension(:), allocatable, public l11_frow
- integer(i4), dimension(:), allocatable, public l11_fcol
- integer(i4), dimension(:), allocatable, public l11_trow
- integer(i4), dimension(:), allocatable, public l11_tcol
- integer(i4), dimension(:), allocatable, public l11_rorder
- integer(i4), dimension(:), allocatable, public l11_label
- logical, dimension(:), allocatable, public l11_sink
- real(dp), dimension(:), allocatable, public l11_length
- real(dp), dimension(:), allocatable, target, public l11_afloodplain
- real(dp), dimension(:), allocatable, public l11_slope
- real(dp), dimension(:, :), allocatable, public l11_nlinkfracfpimp
- real(dp), dimension(:), allocatable, public l11_k
- real(dp), dimension(:), allocatable, public l11_xi
- real(dp), dimension(:), allocatable, public l11_tsrout
- real(dp), dimension(:), allocatable, public l11_c1
- real(dp), dimension(:), allocatable, public l11_c2
- logical gw_coupling
- real(dp), dimension(:), allocatable, public l11_bankfull_runoff_in
- real(dp), dimension(:), allocatable, public l0_channel_depth
- real(dp), dimension(:), allocatable, public l0_channel_elevation
- real(dp), dimension(:), allocatable, public l0_river_head_mon_sum
- real(dp), dimension(:), allocatable, public l0_slope

16.51.1 Detailed Description

Global variables for mRM only.

TODO: add description

Authors

Luis Samaniego, Stephan Thober

Date

Aug 2015

16.51.2 Variable Documentation

16.51.2.1 basin_mrm

```
type(basininfo_mrm), dimension(:), allocatable, target, public mo_mrm_global_variables::basin↩
_mrm
```

Referenced by mo_mrm_init::config_output(), mo_mrm_net_startup::l11_flow_direction(), mo_mrm_net_startup↩ ::l11_link_location(), mo_mrm_net_startup::l11_set_drain_outlet_gauges(), mo_mhm_eval::mhm_eval(), mo↩ _mrm_eval::mrm_eval(), mo_mrm_init::mrm_init(), mo_mrm_mpr::mrm_init_param(), mo_mrm_read_config↩ ::mrm_read_config(), mo_mrm_read_data::mrm_read_l0_data(), mo_mrm_restart::mrm_read_restart_config(), mo_mrm_write::mrm_write(), mo_mrm_restart::mrm_write_restart(), mo_mrm_write::write_configfile(), and mo_↩ mrm_write::write_daily_obs_sim_discharge().

16.51.2.2 dirbankfullrunoff

```
character(256), dimension(:), allocatable, public mo_mrm_global_variables::dirbankfullrunoff
```

Referenced by mo_mrm_read_data::mrm_read_bankfull_runoff(), and mo_mrm_read_config::mrm_read_config().

16.51.2.3 dirgauges

```
character(256), dimension(:), allocatable, public mo_mrm_global_variables::dirgauges
```

Referenced by mo_mrm_init::config_output(), mo_mrm_read_config::mrm_read_config(), mo_write_ascii::write_↩ configfile(), and mo_mrm_write::write_configfile().

16.51.2.4 dirtotalrunoff

```
character(256), dimension(:), allocatable, public mo_mrm_global_variables::dirtotalrunoff
```

Referenced by mo_mrm_read_config::mrm_read_config(), mo_mrm_read_data::mrm_read_total_runoff(), and mo_mrm_write::write_configfile().

16.51.2.5 filenametotalrunoff

```
character(256), public mo_mrm_global_variables::filenametotalrunoff
```

Referenced by mo_mrm_read_config::mrm_read_config(), and mo_mrm_read_data::mrm_read_total_runoff().

16.51.2.6 gauge

```
type(gaugingstation), public mo_mrm_global_variables::gauge
```

Referenced by mo_mrm_objective_function_runoff::extract_runoff(), mo_mrm_read_config::mrm_read_config(), mo_mrm_read_data::mrm_read_discharge(), mo_mrm_write::mrm_write(), mo_mrm_objective_function_runoff↩ ::multi_objective_ae_fdc_lsv_nse_djf(), mo_write_ascii::write_configfile(), mo_mrm_write::write_configfile(), and mo_mrm_write::write_daily_obs_sim_discharge().

16.51.2.7 gw_coupling

```
logical mo_mrm_global_variables::gw_coupling
```

Referenced by mo_mhm_eval::mhm_eval(), mo_mrm_eval::mrm_eval(), mo_mrm_init::mrm_init(), and mo_mrm↩_read_config::mrm_read_config().

16.51.2.8 inflowgauge

```
type(gaugingstation), public mo_mrm_global_variables::inflowgauge
```

Referenced by mo_mhm_eval::mhm_eval(), mo_mrm_eval::mrm_eval(), mo_mrm_read_config::mrm_read_↩config(), mo_mrm_read_data::mrm_read_discharge(), mo_write_ascii::write_configfile(), and mo_mrm_write↩::write_configfile().

16.51.2.9 is_start

```
logical mo_mrm_global_variables::is_start
```

Referenced by mo_mrm_read_config::mrm_read_config(), and mo_mrm_routing::mrm_routing().

16.51.2.10 l0_celerity

```
real(dp), dimension(:), allocatable, public mo_mrm_global_variables::l0_celerity
```

Referenced by mo_mrm_init::variables_alloc_routing().

16.51.2.11 l0_channel_depth

```
real(dp), dimension(:), allocatable, public mo_mrm_global_variables::l0_channel_depth
```

Referenced by mo_mrm_river_head::calc_channel_elevation().

16.51.2.12 l0_channel_elevation

```
real(dp), dimension(:), allocatable, public mo_mrm_global_variables::l0_channel_elevation
```

Referenced by mo_mrm_river_head::calc_river_head().

16.51.2.13 l0_dracell

```
integer(i4), dimension(:), allocatable, public mo_mrm_global_variables::l0_dracell
```

Referenced by mo_mrm_net_startup::l11_set_drain_outlet_gauges().

16.51.2.14 l0_drasc

```
integer(i4), dimension(:), allocatable, public mo_mrm_global_variables::l0_drasc
```

Referenced by mo_mrm_net_startup::l11_flow_direction(), mo_mrm_net_startup::l11_link_location(), and mo_↩ mrm_net_startup::l11_set_drain_outlet_gauges().

16.51.2.15 l0_facc

```
integer(i4), dimension(:), allocatable, public mo_mrm_global_variables::l0_facc
```

Referenced by mo_mrm_river_head::calc_channel_elevation(), mo_mrm_init::l0_check_input_routing(), mo_↩ mrm_net_startup::l11_flow_direction(), mo_mrm_read_data::mrm_read_l0_data(), mo_mrm_restart::mrm_read_↩ restart_config(), and mo_mrm_restart::mrm_write_restart().

16.51.2.16 l0_fdir

```
integer(i4), dimension(:), allocatable, public mo_mrm_global_variables::l0_fdir
```

Referenced by mo_mrm_river_head::calc_channel_elevation(), mo_mrm_init::l0_check_input_routing(), mo_↩ mrm_net_startup::l11_calc_celerity(), mo_mrm_net_startup::l11_flow_direction(), mo_mrm_net_startup::l11_link↩ _location(), mo_mrm_net_startup::l11_set_drain_outlet_gauges(), mo_mrm_net_startup::l11_stream_features(), mo_mrm_read_data::mrm_read_l0_data(), mo_mrm_restart::mrm_read_restart_config(), and mo_mrm_restart↩ ::mrm_write_restart().

16.51.2.17 l0_floodplain

```
integer(i4), dimension(:), allocatable, public mo_mrm_global_variables::l0_floodplain
```

Referenced by mo_mrm_net_startup::l11_fraction_sealed_floodplain(), and mo_mrm_net_startup::l11_stream_↩ features().

16.51.2.18 l0_gaugeloc

```
integer(i4), dimension(:), allocatable, public mo_mrm_global_variables::l0_gaugeloc
```

Referenced by mo_mrm_net_startup::l11_set_drain_outlet_gauges(), and mo_mrm_read_data::mrm_read_l0_↩ data().

16.51.2.19 l0_inflowgaugeloc

```
integer(i4), dimension(:), allocatable, public mo_mrm_global_variables::l0_inflowgaugeloc
```

Referenced by mo_mrm_net_startup::l11_set_drain_outlet_gauges(), and mo_mrm_read_data::mrm_read_l0_↩ data().

16.51.2.20 l0_l11_remap

```
type(gridremapper), dimension(:), allocatable, public mo_mrm_global_variables::l0_l11_remap
```

Referenced by mo_mrm_river_head::calc_river_head(), mo_mrm_net_startup::l11_flow_direction(), mo_mrm_↩ net_startup::l11_set_drain_outlet_gauges(), and mo_mrm_init::mrm_init().

16.51.2.21 l0_noutlet

```
integer(i4), dimension(:), allocatable, public mo_mrm_global_variables::l0_noutlet
```

16.51.2.22 l0_river_head_mon_sum

```
real(dp), dimension(:), allocatable, public mo_mrm_global_variables::l0_river_head_mon_sum
```

Referenced by mo_mhm_eval::mhm_eval(), mo_mrm_eval::mrm_eval(), and mo_mrm_init::mrm_init().

16.51.2.23 l0_slope

```
real(dp), dimension(:), allocatable, public mo_mrm_global_variables::l0_slope
```

Referenced by mo_mrm_river_head::calc_river_head().

16.51.2.24 l0_streamnet

```
integer(i4), dimension(:), allocatable, public mo_mrm_global_variables::l0_streamnet
```

Referenced by mo_mrm_net_startup::l11_stream_features(), mo_mrm_restart::mrm_read_restart_config(), and mo_mrm_restart::mrm_write_restart().

16.51.2.25 l11_afloodplain

```
real(dp), dimension(:), allocatable, target, public mo_mrm_global_variables::l11_afloodplain
```

Referenced by mo_mrm_net_startup::l11_fraction_sealed_floodplain(), mo_mrm_net_startup::l11_stream_↩ features(), mo_mrm_restart::mrm_read_restart_config(), and mo_mrm_restart::mrm_write_restart().

16.51.2.26 l11_areacell

```
real(dp), dimension(:), allocatable, public mo_mrm_global_variables::l11_areacell
```

16.51.2.27 l11_bankfull_runoff_in

```
real(dp), dimension(:), allocatable, public mo_mrm_global_variables::l11_bankfull_runoff_in
```

Referenced by mo_mrm_river_head::calc_river_head().

16.51.2.28 l11_c1

```
real(dp), dimension(:), allocatable, public mo_mrm_global_variables::l11_c1
```

Referenced by mo_mhm_eval::mhm_eval(), mo_mrm_eval::mrm_eval(), mo_mrm_restart::mrm_read_restart_↩ states(), mo_mrm_mpr::mrm_update_param(), mo_mrm_restart::mrm_write_restart(), mo_mrm_init::variables_↩ alloc_routing(), and mo_mrm_init::variables_default_init_routing().

16.51.2.29 l11_c2

```
real(dp), dimension(:), allocatable, public mo_mrm_global_variables::l11_c2
```

Referenced by mo_mhm_eval::mhm_eval(), mo_mrm_eval::mrm_eval(), mo_mrm_restart::mrm_read_restart_↩ states(), mo_mrm_mpr::mrm_update_param(), mo_mrm_restart::mrm_write_restart(), mo_mrm_init::variables_↩ alloc_routing(), and mo_mrm_init::variables_default_init_routing().

16.51.2.30 l11_celerity

```
real(dp), dimension(:), allocatable, public mo_mrm_global_variables::l11_celerity
```

Referenced by mo_mrm_mpr::mrm_init_param(), mo_mrm_mpr::mrm_update_param(), mo_mrm_restart::mrm_↩ write_restart(), and mo_mrm_init::variables_alloc_routing().

16.51.2.31 l11_cellcoor

```
integer(i4), dimension(:,:), allocatable, public mo_mrm_global_variables::l11_cellcoor
```

16.51.2.32 l11_colout

```
integer(i4), dimension(:), allocatable, public mo_mrm_global_variables::l11_colout
```

Referenced by mo_mrm_net_startup::l11_flow_direction(), mo_mrm_net_startup::l11_link_location(), mo_mrm_↩ restart::mrm_read_restart_config(), and mo_mrm_restart::mrm_write_restart().

16.51.2.33 l11_facc

```
real(dp), dimension(:), allocatable, public mo_mrm_global_variables::l11_facc
```

Referenced by mo_mrm_restart::mrm_read_restart_config(), and mo_mrm_restart::mrm_write_restart().

16.51.2.34 l11_fcol

```
integer(i4), dimension(:), allocatable, public mo_mrm_global_variables::l11_fcol
```

Referenced by mo_mrm_net_startup::l11_link_location(), mo_mrm_net_startup::l11_stream_features(), mo_mrm↩
_restart::mrm_read_restart_config(), and mo_mrm_restart::mrm_write_restart().

16.51.2.35 l11_fdir

```
integer(i4), dimension(:), allocatable, public mo_mrm_global_variables::l11_fdir
```

Referenced by mo_mrm_net_startup::l11_flow_accumulation(), mo_mrm_net_startup::l11_flow_direction(), mo↩
_mrm_net_startup::l11_routing_order(), mo_mrm_net_startup::l11_set_network_topology(), mo_mrm_restart↩
::mrm_read_restart_config(), and mo_mrm_restart::mrm_write_restart().

16.51.2.36 l11_fromn

```
integer(i4), dimension(:), allocatable, public mo_mrm_global_variables::l11_fromn
```

Referenced by mo_mrm_net_startup::l11_link_location(), mo_mrm_net_startup::l11_routing_order(), mo_mrm↩
_net_startup::l11_set_network_topology(), mo_mhm_eval::mhm_eval(), mo_mrm_eval::mrm_eval(), mo_mrm_↩
restart::mrm_read_restart_config(), mo_mrm_restart::mrm_write_restart(), mo_write_ascii::write_configfile(), and mo_mrm_write::write_configfile().

16.51.2.37 l11_frow

```
integer(i4), dimension(:), allocatable, public mo_mrm_global_variables::l11_frow
```

Referenced by mo_mrm_net_startup::l11_link_location(), mo_mrm_net_startup::l11_stream_features(), mo_mrm↩
_restart::mrm_read_restart_config(), and mo_mrm_restart::mrm_write_restart().

16.51.2.38 l11_k

```
real(dp), dimension(:), allocatable, public mo_mrm_global_variables::l11_k
```

Referenced by mo_mrm_restart::mrm_read_restart_states(), mo_mrm_restart::mrm_write_restart(), mo_mrm_↩
init::variables_alloc_routing(), and mo_mrm_init::variables_default_init_routing().

16.51.2.39 l11_l1_id

```
integer(i4), dimension(:), allocatable, public mo_mrm_global_variables::l11_l1_id
```

Referenced by mo_mrm_net_startup::l11_l1_mapping(), mo_mhm_eval::mhm_eval(), mo_mrm_eval::mrm_eval(), mo_mrm_restart::mrm_read_restart_config(), mo_mrm_restart::mrm_write_restart(), and mo_mrm_write::write_↩
configfile().

16.51.2.40 l11_label

```
integer(i4), dimension(:), allocatable, public mo_mrm_global_variables::l11_label
```

Referenced by mo_mrm_net_startup::l11_routing_order(), mo_mrm_restart::mrm_read_restart_config(), mo_↩
mrm_restart::mrm_write_restart(), mo_write_ascii::write_configfile(), and mo_mrm_write::write_configfile().

16.51.2.41 l11_length

```
real(dp), dimension(:), allocatable, public mo_mrm_global_variables::l11_length
```

Referenced by mo_mrm_net_startup::l11_stream_features(), mo_mhm_eval::mhm_eval(), mo_mrm_eval::mrm↩
_eval(), mo_mrm_restart::mrm_read_restart_config(), mo_mrm_mpr::mrm_update_param(), mo_mrm_restart↩
::mrm_write_restart(), mo_write_ascii::write_configfile(), and mo_mrm_write::write_configfile().

16.51.2.42 l11_linkin_facc

```
real(dp), dimension(:), allocatable, public mo_mrm_global_variables::l11_linkin_facc
```

16.51.2.43 l11_meandering

```
real(dp), dimension(:), allocatable, public mo_mrm_global_variables::l11_meandering
```

16.51.2.44 l11_netperm

```
integer(i4), dimension(:), allocatable, public mo_mrm_global_variables::l11_netperm
```

Referenced by mo_mrm_net_startup::l11_link_location(), mo_mrm_net_startup::l11_routing_order(), mo_mrm↩
_net_startup::l11_stream_features(), mo_mhm_eval::mhm_eval(), mo_mrm_eval::mrm_eval(), mo_mrm_restart↩
::mrm_read_restart_config(), mo_mrm_restart::mrm_write_restart(), mo_write_ascii::write_configfile(), and mo_↩
mrm_write::write_configfile().

16.51.2.45 l11_nlinkfracfpimp

```
real(dp), dimension(:, :), allocatable, public mo_mrm_global_variables::l11_nlinkfracfpimp
```

Referenced by mo_mrm_net_startup::l11_fraction_sealed_floodplain(), mo_mhm_eval::mhm_eval(), mo_mrm_↩
eval::mrm_eval(), mo_mrm_restart::mrm_read_restart_states(), and mo_mrm_restart::mrm_write_restart().

16.51.2.46 l11_noutlets

```
integer(i4), dimension(:), allocatable, public mo_mrm_global_variables::l11_noutlets
```

Referenced by mo_mrm_net_startup::l11_flow_direction(), mo_mrm_net_startup::l11_fraction_sealed_floodplain(), mo_mrm_net_startup::l11_link_location(), mo_mrm_net_startup::l11_routing_order(), mo_mrm_net_startup::l11↩
_stream_features(), mo_mhm_eval::mhm_eval(), mo_mrm_eval::mrm_eval(), mo_mrm_restart::mrm_read_↩
restart_config(), mo_mrm_mpr::mrm_update_param(), and mo_write_ascii::write_configfile().

16.51.2.47 l11_qmod

```
real(dp), dimension(:), allocatable, public mo_mrm_global_variables::l11_qmod
```

Referenced by mo_mhm_eval::mhm_eval(), mo_mrm_eval::mrm_eval(), mo_mrm_restart::mrm_read_restart↩ _states(), mo_mrm_restart::mrm_write_restart(), mo_mrm_init::variables_alloc_routing(), and mo_mrm_init↩ ::variables_default_init_routing().

16.51.2.48 l11_qout

```
real(dp), dimension(:), allocatable, public mo_mrm_global_variables::l11_qout
```

Referenced by mo_mhm_eval::mhm_eval(), mo_mrm_eval::mrm_eval(), mo_mrm_restart::mrm_read_restart↩ _states(), mo_mrm_restart::mrm_write_restart(), mo_mrm_init::variables_alloc_routing(), and mo_mrm_init↩ ::variables_default_init_routing().

16.51.2.49 l11_qtin

```
real(dp), dimension(:, :), allocatable, public mo_mrm_global_variables::l11_qtin
```

Referenced by mo_mhm_eval::mhm_eval(), mo_mrm_eval::mrm_eval(), mo_mrm_restart::mrm_read_restart↩ _states(), mo_mrm_restart::mrm_write_restart(), mo_mrm_init::variables_alloc_routing(), and mo_mrm_init↩ ::variables_default_init_routing().

16.51.2.50 l11_qtr

```
real(dp), dimension(:, :), allocatable, public mo_mrm_global_variables::l11_qtr
```

Referenced by mo_mhm_eval::mhm_eval(), mo_mrm_eval::mrm_eval(), mo_mrm_restart::mrm_read_restart↩ _states(), mo_mrm_restart::mrm_write_restart(), mo_mrm_init::variables_alloc_routing(), and mo_mrm_init↩ ::variables_default_init_routing().

16.51.2.51 l11_rorder

```
integer(i4), dimension(:), allocatable, public mo_mrm_global_variables::l11_rorder
```

Referenced by mo_mrm_net_startup::l11_routing_order(), mo_mrm_restart::mrm_read_restart_config(), mo_↩ mrm_restart::mrm_write_restart(), mo_write_ascii::write_configfile(), and mo_mrm_write::write_configfile().

16.51.2.52 l11_rowout

```
integer(i4), dimension(:), allocatable, public mo_mrm_global_variables::l11_rowout
```

Referenced by mo_mrm_net_startup::l11_flow_direction(), mo_mrm_net_startup::l11_link_location(), mo_mrm_↩ restart::mrm_read_restart_config(), and mo_mrm_restart::mrm_write_restart().

16.51.2.53 l11_sink

```
logical, dimension(:), allocatable, public mo_mrm_global_variables::l11_sink
```

Referenced by mo_mrm_net_startup::l11_routing_order(), mo_mrm_restart::mrm_read_restart_config(), and mo↩ _mrm_restart::mrm_write_restart().

16.51.2.54 l11_slope

```
real(dp), dimension(:), allocatable, public mo_mrm_global_variables::l11_slope
```

Referenced by mo_mrm_net_startup::l11_stream_features(), mo_mhm_eval::mhm_eval(), mo_mrm_eval::mrm↩
_eval(), mo_mrm_restart::mrm_read_restart_config(), mo_mrm_restart::mrm_write_restart(), mo_write_ascii↩
::write_configfile(), and mo_mrm_write::write_configfile().

16.51.2.55 l11_tcol

```
integer(i4), dimension(:), allocatable, public mo_mrm_global_variables::l11_tcol
```

Referenced by mo_mrm_net_startup::l11_link_location(), mo_mrm_net_startup::l11_stream_features(), mo_mrm↩
_restart::mrm_read_restart_config(), and mo_mrm_restart::mrm_write_restart().

16.51.2.56 l11_ton

```
integer(i4), dimension(:), allocatable, public mo_mrm_global_variables::l11_ton
```

Referenced by mo_mrm_net_startup::l11_routing_order(), mo_mrm_net_startup::l11_set_network_topology(), mo_mhm_eval::mhm_eval(), mo_mrm_eval::mrm_eval(), mo_mrm_restart::mrm_read_restart_config(), mo_mrm↩
_restart::mrm_write_restart(), mo_write_ascii::write_configfile(), and mo_mrm_write::write_configfile().

16.51.2.57 l11_trow

```
integer(i4), dimension(:), allocatable, public mo_mrm_global_variables::l11_trow
```

Referenced by mo_mrm_net_startup::l11_link_location(), mo_mrm_net_startup::l11_stream_features(), mo_mrm↩
_restart::mrm_read_restart_config(), and mo_mrm_restart::mrm_write_restart().

16.51.2.58 l11_tsrout

```
real(dp), dimension(:), allocatable, public mo_mrm_global_variables::l11_tsrout
```

Referenced by mo_mhm_eval::mhm_eval(), mo_mrm_eval::mrm_eval(), mo_mrm_mpr::mrm_init_param(), mo↩
_mrm_restart::mrm_read_restart_config(), mo_mrm_mpr::mrm_update_param(), and mo_mrm_restart::mrm_↩
write_restart().

16.51.2.59 l11_xi

```
real(dp), dimension(:), allocatable, public mo_mrm_global_variables::l11_xi
```

Referenced by mo_mrm_restart::mrm_read_restart_states(), mo_mrm_restart::mrm_write_restart(), mo_mrm_↩
init::variables_alloc_routing(), and mo_mrm_init::variables_default_init_routing().

16.51.2.60 l1_l11_id

```
integer(i4), dimension(:), allocatable, public mo_mrm_global_variables::l1_l11_id
```

Referenced by mo_mrm_net_startup::l11_l1_mapping(), mo_mhm_eval::mhm_eval(), mo_mrm_eval::mrm_eval(), mo_mrm_restart::mrm_read_restart_config(), mo_mrm_restart::mrm_write_restart(), mo_write_ascii::write_configfile(), and mo_mrm_write::write_configfile().

16.51.2.61 l1_l11_remap

```
type(gridremapper), dimension(:), allocatable, public mo_mrm_global_variables::l1_l11_remap
```

Referenced by mo_mrm_init::mrm_init().

16.51.2.62 l1_total_runoff_in

```
real(dp), dimension(:, :), allocatable, public mo_mrm_global_variables::l1_total_runoff_in
```

Referenced by mo_mrm_eval::mrm_eval(), and mo_mrm_read_data::mrm_read_total_runoff().

16.51.2.63 level11

```
type(grid), dimension(:), allocatable, target, public mo_mrm_global_variables::level11
```

Referenced by mo_mrm_write_fluxes_states::createoutputfile(), mo_mrm_net_startup::l11_flow_accumulation(), mo_mrm_net_startup::l11_flow_direction(), mo_mrm_net_startup::l11_fraction_sealed_floodplain(), mo_mrm_net_startup::l11_l1_mapping(), mo_mrm_net_startup::l11_link_location(), mo_mrm_net_startup::l11_routing_order(), mo_mrm_net_startup::l11_set_network_topology(), mo_mrm_net_startup::l11_stream_features(), mo_mhm_eval::mhm_eval(), mo_mrm_eval::mrm_eval(), mo_mrm_init::mrm_init(), mo_mrm_mpr::mrm_init_param(), mo_mrm_read_data::mrm_read_bankfull_runoff(), mo_mrm_restart::mrm_read_restart_config(), mo_mrm_restart::mrm_read_restart_states(), mo_mrm_mpr::mrm_update_param(), mo_mrm_restart::mrm_write_restart(), mo_mrm_init::variables_alloc_routing(), mo_write_ascii::write_configfile(), and mo_mrm_write::write_configfile().

16.51.2.64 mrm_runoff

```
real(dp), dimension(:, :), allocatable, public mo_mrm_global_variables::mrm_runoff
```

Referenced by mo_mhm_eval::mhm_eval(), mo_mrm_eval::mrm_eval(), mo_mrm_read_data::mrm_read_discharge(), and mo_mrm_write::mrm_write().

16.51.2.65 ngaugestotal

```
integer(i4), public mo_mrm_global_variables::ngaugestotal
```

Referenced by mo_mrm_read_config::mrm_read_config(), mo_mrm_read_data::mrm_read_discharge(), mo_mrm_write::mrm_write(), mo_write_ascii::write_configfile(), and mo_mrm_write::write_configfile().

16.51.2.66 ninflowgaugestotal

```
integer(i4), public mo_mrm_global_variables::ninflowgaugestotal
```

Referenced by mo_mrm_read_config::mrm_read_config(), mo_mrm_read_data::mrm_read_discharge(), mo_write_ascii::write_configfile(), and mo_mrm_write::write_configfile().

16.51.2.67 nmeasperday

```
integer(i4), public mo_mrm_global_variables::nmeasperday
```

Referenced by mo_mrm_objective_function_runoff::extract_runoff(), mo_mrm_read_data::mrm_read_discharge(), and mo_mrm_objective_function_runoff::multi_objective_ae_fdc_lsv_nse_djf().

16.51.2.68 ntstepday

```
integer(i4), public mo_mrm_global_variables::ntstepday
```

16.51.2.69 outputflxstate_mrm

```
logical, dimension(noutflxstate) mo_mrm_global_variables::outputflxstate_mrm
```

Referenced by mo_mhm_eval::mhm_eval(), mo_mrm_eval::mrm_eval(), mo_mrm_read_config::mrm_read_config(), mo_mrm_write_fluxes_states::newoutputdataset(), and mo_mrm_write_fluxes_states::updatedataset().

16.51.2.70 timestep_model_outputs_mrm

```
integer(i4) mo_mrm_global_variables::timestep_model_outputs_mrm
```

Referenced by mo_mhm_eval::mhm_eval(), mo_mrm_eval::mrm_eval(), and mo_mrm_read_config::mrm_read_config().

16.51.2.71 varnametotalrunoff

```
character(256), public mo_mrm_global_variables::varnametotalrunoff
```

Referenced by mo_mrm_read_config::mrm_read_config(), and mo_mrm_read_data::mrm_read_total_runoff().

16.52 mo_mrm_init Module Reference

Wrapper for initializing Routing.

Functions/Subroutines

- subroutine, public mrm_init (file_namelist, unamelist, file_namelist_param, unamelist_param)

  Initialize all mRM variables at all levels (i.e., L0, L1, and L11).
- subroutine print_startup_message (file_namelist, file_namelist_param)

TODO: add description.

- subroutine config_output

  TODO: add description.

- subroutine, public variables_default_init_routing

  Default initalization mRM related L11 variables.

- subroutine l0_check_input_routing (L0Basin_iBasin)

  TODO: add description.

- subroutine variables_alloc_routing (iBasin)

  TODO: add description.

16.52.1 Detailed Description

Wrapper for initializing Routing.

Calling all routines to initialize all mRM variables

Authors

Luis Samaniego, Rohini Kumar and Stephan Thober

Date

Aug 2015

16.52.2 Function/Subroutine Documentation

16.52.2.1 config_output()

```
subroutine mo_mrm_init::config_output ( )
```

TODO: add description.

TODO: add description

Authors

Robert Schweppe

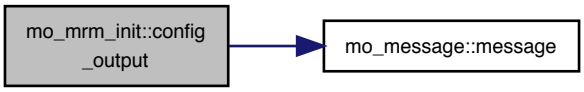
Date

Jun 2018

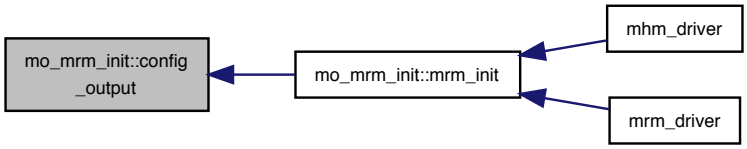
References mo_mrm_global_variables::basin_mrm, mo_mrm_global_variables::dirgauges, mo_common_↩ variables::dirlcover, mo_common_variables::dirmorpho, mo_common_variables::dirout, mo_mrm_file::file_↩ defoutput, mo_mrm_file::file_namelist_mrm, mo_mrm_file::file_namelist_param_mrm, mo_kind::i4, mo_message↩ ::message(), and mo_common_variables::nbasins.

Referenced by mrm_init().

Here is the call graph for this function:



Here is the caller graph for this function:



16.52.2.2 l0_check_input_routing()

```
subroutine mo_mrm_init::l0_check_input_routing (
             integer(i4), intent(in) L0Basin_iBasin )
```

TODO: add description.

TODO: add description

Parameters

in	*integer(i4) :: L0Basin_iBasin*	

Authors

Robert Schweppe

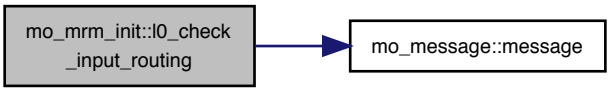
Date

Jun 2018

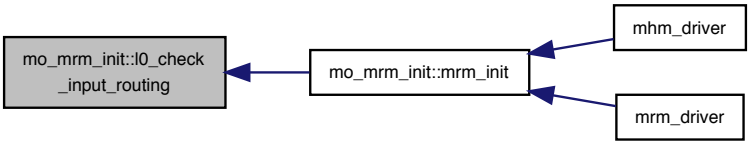
References mo_kind::i4, mo_mrm_global_variables::l0_facc, mo_mrm_global_variables::l0_fdir, mo_common_↩ variables::level0, mo_message::message(), mo_message::message_text, and mo_common_constants::nodata_i4.

Referenced by mrm_init().

Here is the call graph for this function:



Here is the caller graph for this function:



16.52.2.3 mrm_init()

```
subroutine, public mo_mrm_init::mrm_init (
             character(*), intent(in) file_namelist,
             integer, intent(in) unamelist,
             character(*), intent(in) file_namelist_param,
             integer, intent(in) unamelist_param )
```

Initialize all mRM variables at all levels (i.e., L0, L1, and L11).

Initialize all mRM variables at all levels (i.e., L0, L1, and L11) either with default values or with values from restart file. The L0 mask (L0_mask), L0 elevation (L0_elev), and L0 land cover (L0_LCover) can be provided as optional variables to save memory because these variable will then not be read in again.

Parameters

in	*character(∗) :: file_namelist, file_namelist_param*	
in	*integer :: unamelist, unamelist_param*	
in	*character(∗) :: file_namelist, file_namelist_param*	
in	*integer :: unamelist, unamelist_param*	

Authors

Stephan Thober

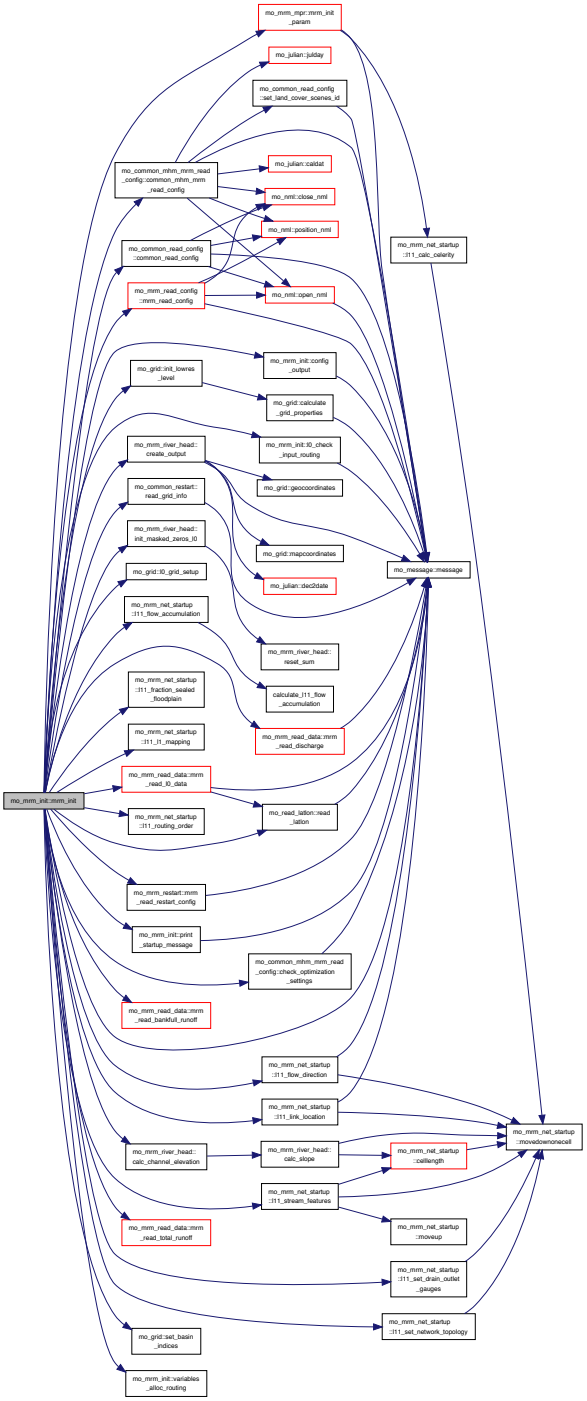
Date

Aug 2015

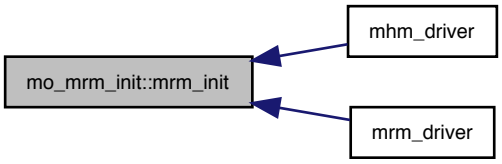
References mo_mrm_global_variables::basin_mrm, mo_mrm_river_head::calc_channel_elevation(), mo_common_mhm_mrm_read_config::check_optimization_settings(), mo_common_mhm_mrm_read_config::common_mhm_mrm_read_config(), mo_common_read_config::common_read_config(), config_output(), mo_mrm_river_head::create_output(), mo_common_mhm_mrm_variables::dirrestartin, mo_common_variables::global_parameters, mo_mrm_global_variables::gw_coupling, mo_kind::i4, mo_grid::init_lowres_level(), mo_mrm_river_head::init_masked_zeros_l0(), mo_common_variables::l0_basin, l0_check_input_routing(), mo_grid::l0_grid_setup(), mo_mrm_global_variables::l0_l11_remap, mo_common_variables::l0_l1_remap, mo_mrm_global_variables::l0_river_head_mon_sum, mo_mrm_net_startup::l11_flow_accumulation(), mo_mrm_net_startup::l11_flow_direction(), mo_mrm_net_startup::l11_fraction_sealed_floodplain(), mo_mrm_net_startup::l11_l1_mapping(), mo_mrm_net_startup::l11_link_location(), mo_mrm_net_startup::l11_routing_order(), mo_mrm_net_startup::l11_set_drain_outlet_gauges(), mo_mrm_net_startup::l11_set_network_topology(), mo_mrm_net_startup::l11_stream_features(), mo_mrm_global_variables::l1_l11_remap, mo_common_variables::level0, mo_common_variables::level1, mo_mrm_global_variables::level11, mo_message::message(), mo_common_mhm_mrm_variables::mrm_coupling_mode, mo_mrm_mpr::mrm_init_param(), mo_mrm_read_data::mrm_read_bankfull_runoff(), mo_mrm_read_config::mrm_read_config(), mo_mrm_read_data::mrm_read_discharge(), mo_mrm_read_data::mrm_read_l0_data(), mo_mrm_restart::mrm_read_restart_config(), mo_mrm_read_data::mrm_read_total_runoff(), mo_common_variables::nbasins, mo_common_constants::nodata_dp, mo_common_constants::nodata_i4, print_startup_message(), mo_common_variables::processmatrix, mo_common_restart::read_grid_info(), mo_read_latlon::read_latlon(), mo_common_mhm_mrm_variables::read_restart, mo_common_variables::resolutionhydrology, mo_common_mhm_mrm_variables::resolutionrouting, mo_grid::set_basin_indices(), and variables_alloc_routing().

Referenced by mhm_driver(), and mrm_driver().

Here is the call graph for this function:



Here is the caller graph for this function:



16.52.2.4 print_startup_message()

```
subroutine mo_mrm_init::print_startup_message (
             character(*), intent(in) file_namelist,
             character(*), intent(in) file_namelist_param )
```

TODO: add description.

TODO: add description

Parameters

in	*character(*) :: file_namelist, file_namelist_param*	
in	*character(*) :: file_namelist, file_namelist_param*	

Authors

Robert Schweppe

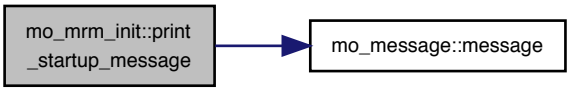
Date

Jun 2018

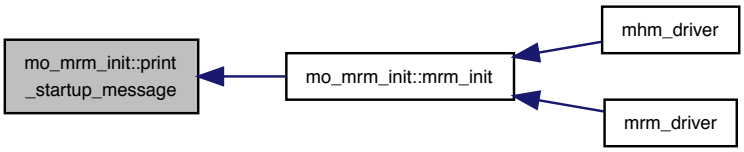
References mo_mrm_file::file_defoutput, mo_mrm_file::file_main, mo_kind::i4, mo_message::message(), mo_↩ message::message_text, mo_string_utils::separator, mo_mrm_file::version, and mo_mrm_file::version_date.

Referenced by mrm_init().

Here is the call graph for this function:



Here is the caller graph for this function:



16.52.2.5 variables_alloc_routing()

```
subroutine mo_mrm_init::variables_alloc_routing (
             integer(i4), intent(in) iBasin )
```

TODO: add description.

TODO: add description

Parameters

in	*integer(i4) :: iBasin*	

Authors

Robert Schweppe

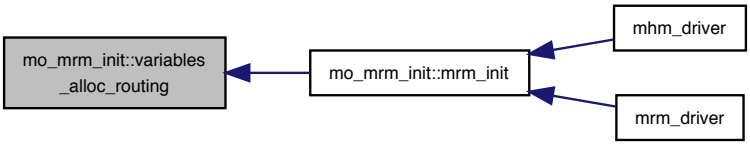
Date

Jun 2018

References mo_kind::dp, mo_kind::i4, mo_common_variables::l0_basin, mo_mrm_global_variables::l0_celerity, mo_mrm_global_variables::l11_c1, mo_mrm_global_variables::l11_c2, mo_mrm_global_variables::l11_celerity, mo_mrm_global_variables::l11_k, mo_mrm_global_variables::l11_qmod, mo_mrm_global_variables::l11_qout, mo_mrm_global_variables::l11_qtin, mo_mrm_global_variables::l11_qtr, mo_mrm_global_variables::l11_xi, mo_↩common_variables::level0, mo_mrm_global_variables::level11, and mo_mrm_constants::nroutingstates.

Referenced by mrm_init().

Here is the caller graph for this function:



16.52.2.6 variables_default_init_routing()

```
subroutine, public mo_mrm_init::variables_default_init_routing ( )
```

Default initalization mRM related L11 variables.

Default initalization of mHM related L11 variables (e.g., states, fluxes, and parameters) as per given constant values given in mo_mhm_constants. Variables initalized here is defined in the mo_global_variables.f90 file. Only Variables that are defined in the variables_alloc subroutine are intialized here. If a variable is added or removed here, then it also has to be added or removed in the subroutine state_variables_set in the module mo_restart and in the subroutine set_state in the module mo_set_netcdf_restart. ADDITIONAL INFORMATION variables_default_init_routing call variables_default_init()

Authors

Stephan Thober, Rohini Kumar, and Juliane Mai

Date

Aug 2015

Authors

Robert Schweppe

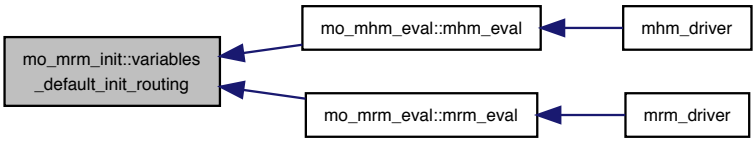
Date

Jun 2018

References mo_mrm_global_variables::l11_c1, mo_mrm_global_variables::l11_c2, mo_mrm_global_variables::l11_k, mo_mrm_global_variables::l11_qmod, mo_mrm_global_variables::l11_qout, mo_mrm_global_variables::l11_qtin, mo_mrm_global_variables::l11_qtr, mo_mrm_global_variables::l11_xi, and mo_common_constants::p1_initstatefluxes.

Referenced by mo_mhm_eval::mhm_eval(), and mo_mrm_eval::mrm_eval().

Here is the caller graph for this function:



16.53 mo_mrm_mpr Module Reference

Perform Multiscale Parameter Regionalization on Routing Parameters.

Functions/Subroutines

- subroutine, public reg_rout (param, length, slope, fFPimp, TS, C1, C2)

  Regionalized routing.
- subroutine, public mrm_init_param (iBasin, param)

  TODO: add description.
- subroutine, public mrm_update_param (iBasin, param)

  TODO: add description.

16.53.1 Detailed Description

Perform Multiscale Parameter Regionalization on Routing Parameters.

This module contains the subroutine for calculating the regionalized routing parameters (beta-parameters) given the five global routing parameters (gamma) at the level 0 scale.

Authors

Luis Samaniego, Stephan Thober

Date

Aug 2015

16.53.2 Function/Subroutine Documentation

16.53.2.1 mrm_init_param()

```
subroutine, public mo_mrm_mpr::mrm_init_param (
             integer(i4), intent(in) iBasin,
             real(dp), dimension(:), intent(in) param )
```

TODO: add description.

TODO: add description

Parameters

in	*integer(i4) :: iBasin*	Basin number
in	*real(dp), dimension(:) :: param*	input parameter (param(1) is celerity in m/s)

Authors

Stephan Thober, Matthias Kelbling

Date

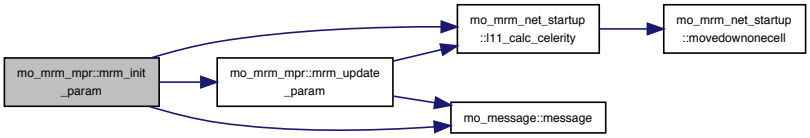
Jun 2018

References mo_mrm_global_variables::basin_mrm, mo_kind::dp, mo_mrm_constants::given_ts, mo_common↩ _constants::hoursecs, mo_kind::i4, mo_common_variables::iflag_cordinate_sys, mo_mrm_net_startup::l11_↩ calc_celerity(), mo_mrm_global_variables::l11_celerity, mo_mrm_global_variables::l11_tsrout, mo_mrm_global↩ _variables::level11, mo_message::message(), mrm_update_param(), mo_common_variables::nbasins, mo_↩

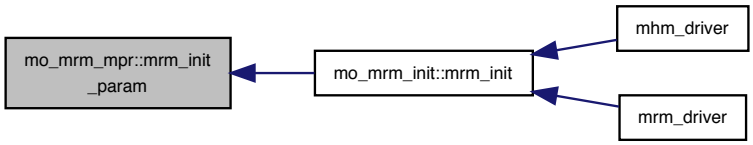
common_mhm_mrm_variables::optimize, mo_common_variables::processmatrix, mo_common_mhm_mrm_↩ variables::resolutionrouting, and mo_common_mhm_mrm_variables::timestep.

Referenced by mo_mrm_init::mrm_init().

Here is the call graph for this function:



Here is the caller graph for this function:



16.53.2.2 mrm_update_param()

```
subroutine, public mo_mrm_mpr::mrm_update_param (
             integer(i4), intent(in) iBasin,
             real(dp), dimension(1), intent(in) param )
```

TODO: add description.

TODO: add description

Parameters

in	*integer(i4) :: iBasin*	Basin number
in	*real(dp), dimension(1) :: param*	celerity parameter [m s-1]

Authors

Stehpan Thober, Matthias Kelbling

Date

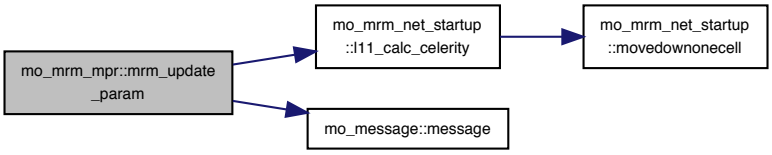
Jun 2018

References mo_kind::dp, mo_mrm_constants::given_ts, mo_common_constants::hoursecs, mo_kind::i4, mo_↩ common_variables::iflag_cordinate_sys, mo_mrm_global_variables::l11_c1, mo_mrm_global_variables::l11_c2,

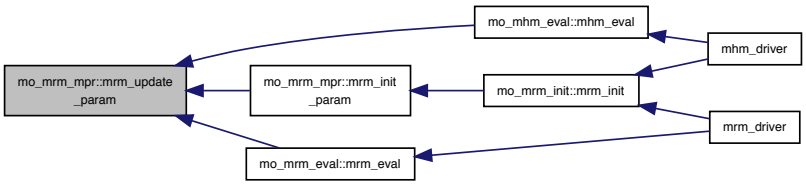
mo_mrm_net_startup::l11_calc_celerity(), mo_mrm_global_variables::l11_celerity, mo_mrm_global_variables::l11_length, mo_mrm_global_variables::l11_noutlets, mo_mrm_global_variables::l11_tsrout, mo_mrm_global_variables::level11, mo_message::message(), mo_common_mhm_mrm_variables::optimize, mo_common_variables::processmatrix, mo_common_mhm_mrm_variables::resolutionrouting, mo_mrm_constants::rout_space_weight, and mo_common_mhm_mrm_variables::timestep.

Referenced by mo_mhm_eval::mhm_eval(), mo_mrm_eval::mrm_eval(), and mrm_init_param().

Here is the call graph for this function:



Here is the caller graph for this function:



16.53.2.3 reg_rout()

```
subroutine, public mo_mrm_mpr::reg_rout (
             real(dp), dimension(5), intent(in) param,
             real(dp), dimension(:), intent(in) length,
             real(dp), dimension(:), intent(in) slope,
             real(dp), dimension(:), intent(in) fFPimp,
             real(dp), intent(in) TS,
             real(dp), dimension(:), intent(out) C1,
             real(dp), dimension(:), intent(out) C2 )
```

Regionalized routing.

sets up the Regionalized Routing parameters Global parameters needed (see mhm_parameter.nml):

- param(1) = muskingumTravelTime_constant
- param(2) = muskingumTravelTime_riverLength
- param(3) = muskingumTravelTime_riverSlope
- param(4) = muskingumTravelTime_impervious

- param(5) = muskingumAttenuation_riverSlope

Parameters

in	*real(dp), dimension(5) :: param*	input parameter
in	*real(dp), dimension(:) :: length*	[m] total length
in	*real(dp), dimension(:) :: slope*	average slope
in	*real(dp), dimension(:) :: fFPimp*	fraction of the flood plain with impervious layer
in	*real(dp) :: TS*	- [h] time step in
out	*real(dp), dimension(:) :: C1*	routing parameter C1 (Chow, 25-41)
out	*real(dp), dimension(:) :: C2*	routing parameter C2 (")

Authors

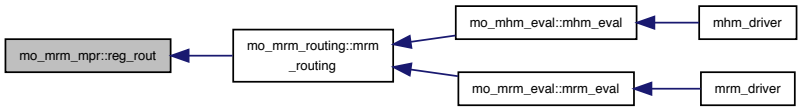
Stephan Thober, Rohini Kumar

Date

Dec 2012

Referenced by mo_mrm_routing::mrm_routing().

Here is the caller graph for this function:



16.54 mo_mrm_net_startup Module Reference

Startup drainage network for mHM.

Functions/Subroutines

- subroutine, public l11_l1_mapping (iBasin)

  TODO: add description.
- subroutine, public l11_flow_direction (iBasin)

  Determine the flow direction of the upscaled river network at level L11.
- subroutine, public l11_set_network_topology (iBasin)

  Set network topology.
- subroutine, public l11_routing_order (iBasin)

  Find routing order, headwater cells and sink.
- subroutine, public l11_link_location (iBasin)

  Estimate the LO (row,col) location for each routing link at level L11.
- subroutine, public l11_set_drain_outlet_gauges (iBasin)

  Draining cell identification and Set gauging node.
- subroutine, public l11_stream_features (iBasin)

  Stream features (stream network and floodplain)
- subroutine, public l11_fraction_sealed_floodplain (LCClassImp, do_init)

Fraction of the flood plain with impervious cover.

- subroutine moveup (elev0, fDir0, fi, fj, ss, nn)

  TODO: add description.
- subroutine movedownonecell (fDir, iRow, jCol)

  TODO: add description.
- subroutine celllength (iBasin, fDir, iRow, jCol, iCoorSystem, length)

  TODO: add description.
- subroutine, public get_distance_two_lat_lon_points (lat1, long1, lat2, long2, distance_out)

  estimate distance in [m] between two points in a lat-lon
- subroutine, public l11_flow_accumulation (iBasin)

  Calculates L11 flow accumulation per grid cell.
- subroutine, public l11_calc_celerity (iBasin, param)

  L11 celerity based on L0 elevation and L0 fAcc.

16.54.1 Detailed Description

Startup drainage network for mHM.

This module initializes the drainage network at L11 in mHM.

- Delineation of drainage network at level 11.
- Setting network topology (i.e. nodes and link).
- Determining routing order.
- Determining cell locations for network links.
- Find drainage outlet.
- Determine stream (links) features.

Authors

Luis Samaniego

Date

Dec 2012

16.54.2 Function/Subroutine Documentation

16.54.2.1 celllength()

```
subroutine mo_mrm_net_startup::celllength (
             integer(i4), intent(in) iBasin,
             integer(i4), intent(in) fDir,
             integer(i4), intent(in) iRow,
             integer(i4), intent(in) jCol,
             integer(i4), intent(in) iCoorSystem,
             real(dp), intent(out) length )
```

TODO: add description.

TODO: add description

Parameters

in	*integer(i4) :: iBasin*	
in	*integer(i4) :: fDir*	
in	*integer(i4) :: iRow*	
in	*integer(i4) :: jCol*	
in	*integer(i4) :: iCoorSystem*	
out	*real(dp) :: length*	

Authors

Robert Schweppe

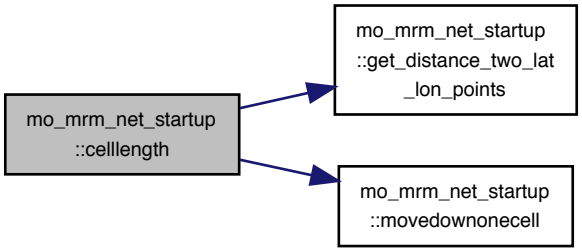
Date

Jun 2018

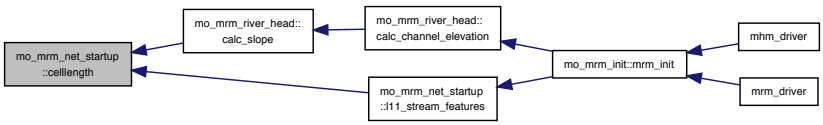
References get_distance_two_lat_lon_points(), mo_common_variables::l0_basin, mo_common_variables::level0, movedownonecell(), and mo_constants::sqrt2_dp.

Referenced by mo_mrm_river_head::calc_slope(), and l11_stream_features().

Here is the call graph for this function:



Here is the caller graph for this function:



16.54.2.2 get_distance_two_lat_lon_points()

```
subroutine, public mo_mrm_net_startup::get_distance_two_lat_lon_points (
             real(dp), intent(in) lat1,
```

```
            real(dp), intent(in) long1,
            real(dp), intent(in) lat2,
            real(dp), intent(in) long2,
            real(dp), intent(out) distance_out )
```

estimate distance in [m] between two points in a lat-lon

estimate distance in [m] between two points in a lat-lon Code is based on one that is implemented in the VIC-3L model

Parameters

in	*real(dp) :: lat1, long1, lat2, long2*	latitude of point-1
in	*real(dp) :: lat1, long1, lat2, long2*	longitude of point-1
in	*real(dp) :: lat1, long1, lat2, long2*	latitude of point-2
in	*real(dp) :: lat1, long1, lat2, long2*	longitude of point-2
out	*real(dp) :: distance_out*	distance between two points [m]

Authors

Rohini Kumar

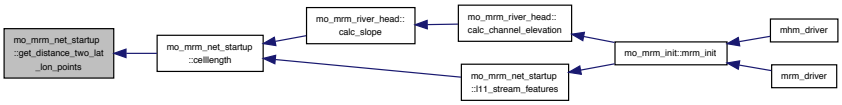
Date

May 2014

References mo_constants::radiusearth_dp, and mo_constants::twopi_dp.

Referenced by celllength().

Here is the caller graph for this function:



16.54.2.3 l11_calc_celerity()

```
subroutine, public mo_mrm_net_startup::l11_calc_celerity (
            integer(i4), intent(in) iBasin,
            real(dp), dimension(:), intent(in) param )
```

L11 celerity based on L0 elevation and L0 fAcc.

L11 celerity based on L0 elevation and L0 fAcc

Parameters

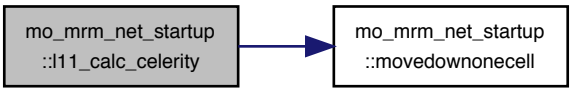
in		

References mo_common_variables::l0_basin, mo_mrm_global_variables::l0_fdir, mo_mpr_global_variables::l0_↩

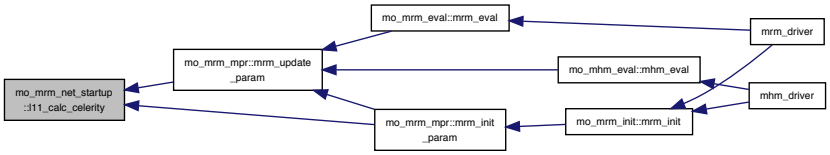
slope, movedownonecell(), mo_common_constants::nodata_dp, and mo_common_constants::nodata_i4.

Referenced by mo_mrm_mpr::mrm_init_param(), and mo_mrm_mpr::mrm_update_param().

Here is the call graph for this function:



Here is the caller graph for this function:



16.54.2.4 l11_flow_accumulation()

```
subroutine, public mo_mrm_net_startup::l11_flow_accumulation (
             integer(i4), intent(in) iBasin )
```

Calculates L11 flow accumulation per grid cell.

Calculates L11 flow accumulation per grid cell using L11_fDir and L11_cellarea. L11_flow_accumulation contains the recursiv subroutine calculate_L11_flow_accumulation iBasin

Author

Matthias Kelbling

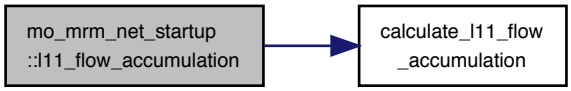
Date

Aug 2017

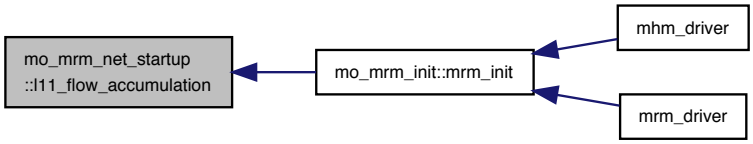
References calculate_l11_flow_accumulation(), mo_mrm_global_variables::l11_fdir, mo_mrm_global_variables::level11, mo_common_constants::nodata_dp, and mo_common_constants::nodata_i4.

Referenced by mo_mrm_init::mrm_init().

Here is the call graph for this function:



Here is the caller graph for this function:



16.54.2.5 l11_flow_direction()

```
subroutine, public mo_mrm_net_startup::l11_flow_direction (
             integer(i4), intent(in) iBasin )
```

Determine the flow direction of the upscaled river network at level L11.

The hydrographs generated at each cell are routed through the drainage network at level-11 towards their outlets. The drainage network at level-11 is conceptualized as a graph whose nodes are hypothetically located at the center of each grid cell connected by links that represent the river reaches. The flow direction of a link correspond to the direction towards a neighboring cell in which the net flow accumulation (outflows minus inflows) attains its maximum value. The net flow accumulation across a cell's boundary at level-11 is estimated based on flow direction and flow accumulation obtained at level-0 (Routing Network"). Note: level-1 denotes the modeling level, whereas level-L11 is at least as coarse as level-1. Experience has shown that routing can be done at a coarser resolution as level-1, hence the level-11 was introduced.

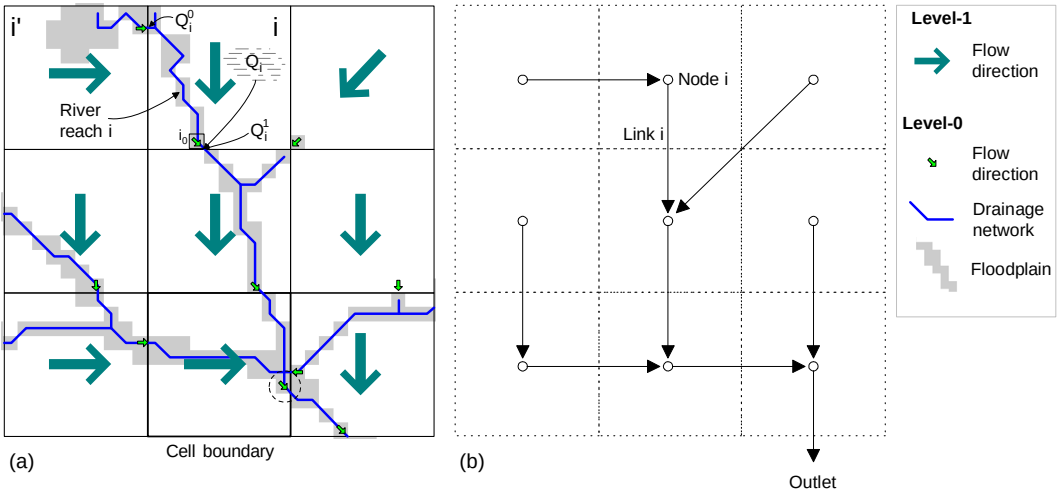


Figure 16.1: Upscaling routing network from L0 to L1 (or L11)

The left panel depicts a schematic derivation of a drainage network at the level-11 based on level-0 flow direction and flow accumulation. The dotted line circle denotes the point with the highest flow accumulation within a grid cell. The topology of a tipical drainage routing network at level-11 is shown in the right panel. Gray color areas denote the flood plains estimated in mo_init_mrm, where the network upscaling is also carried out. For the sake of simplicity, it is assumed that all runoff leaving a given cell would exit through a major direction. Note that multiple outlets can exist within the modelling domain. If a variable is added or removed here, then it also has to be added or removed in the subroutine L11_config_set in module mo_restart and in the subroutine set_L11_config in module mo_set_netcdf_restart ADDITIONAL INFORMATION L11_flow_direction

Parameters

in	*integer(i4) :: iBasin*	Basin Id

Authors

Luis Samaniego

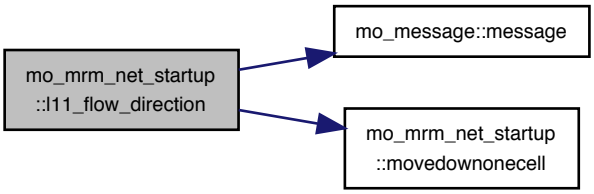
Date

Dec 2005

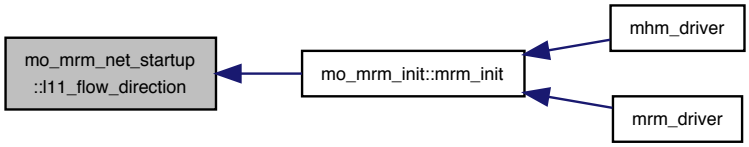
References mo_mrm_global_variables::basin_mrm, mo_common_variables::l0_basin, mo_mrm_global_↩ variables::l0_drasc, mo_mrm_global_variables::l0_facc, mo_mrm_global_variables::l0_fdir, mo_mrm_global_↩ variables::l0_l11_remap, mo_mrm_global_variables::l11_colout, mo_mrm_global_variables::l11_fdir, mo_mrm_↩ global_variables::l11_noutlets, mo_mrm_global_variables::l11_rowout, mo_common_variables::level0, mo_mrm↩ _global_variables::level11, mo_message::message(), movedownonecell(), and mo_common_constants::nodata_i4.

Referenced by mo_mrm_init::mrm_init().

Here is the call graph for this function:



Here is the caller graph for this function:



16.54.2.6 l11_fraction_sealed_floodplain()

```
subroutine, public mo_mrm_net_startup::l11_fraction_sealed_floodplain (
             integer(i4), intent(in) LCClassImp,
             logical, intent(in) do_init )
```

Fraction of the flood plain with impervious cover.

Fraction of the flood plain with impervious cover (Routing Network"). This proportion is used to regionalize the Muskingum parameters. Samaniego et al. [15] found out that this fraction is one of the statistically significant predictor variables of peak discharge in mesoscale basins. If a variable is added or removed here, then it also has to be added or removed in the subroutine L11_config_set in module mo_restart and in the subroutine set_L11_↩ config in module mo_set_netcdf_restart

Parameters

in	*integer(i4) :: LCClassImp*	Impervious land cover class Id, e.g. = 2 (old code)
in	*logical :: do_init*	

Authors

Luis Samaniego

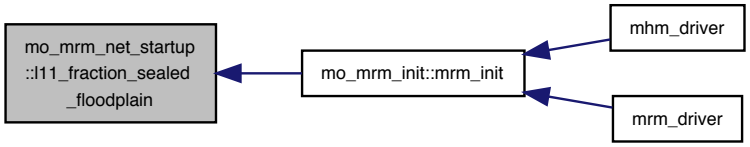
Date

Dec 2005

References mo_common_variables::l0_basin, mo_mrm_global_variables::l0_floodplain, mo_common_variables::l0_lcover, mo_mrm_global_variables::l11_afloodplain, mo_mrm_global_variables::l11_nlinkfracfpimp, mo_mrm_global_variables::l11_noutlets, mo_common_variables::level0, mo_mrm_global_variables::level11, mo_common_variables::nbasins, mo_common_variables::nlcoverscene, and mo_common_constants::nodata_dp.

Referenced by mo_mrm_init::mrm_init().

Here is the caller graph for this function:



16.54.2.7 l11_l1_mapping()

```
subroutine, public mo_mrm_net_startup::l11_l1_mapping (
             integer(i4), intent(in) iBasin )
```

TODO: add description.

TODO: add description

Parameters

in	*integer(i4) :: iBasin*	basin

Authors

Robert Schweppe

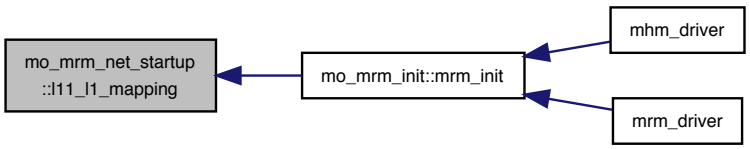
Date

Jun 2018

References mo_kind::i4, mo_mrm_global_variables::l11_l1_id, mo_mrm_global_variables::l1_l11_id, mo_common_variables::level1, mo_mrm_global_variables::level11, and mo_common_constants::nodata_i4.

Referenced by mo_mrm_init::mrm_init().

Here is the caller graph for this function:



16.54.2.8 l11_link_location()

```
subroutine, public mo_mrm_net_startup::l11_link_location (
             integer(i4), intent(in) iBasin )
```

Estimate the LO (row,col) location for each routing link at level L11.

If a variable is added or removed here, then it also has to be added or removed in the subroutine L11_config_set in module mo_restart and in the subroutine set_L11_config in module mo_set_netcdf_restart

Parameters

in	*integer(i4) :: iBasin*	Basin Id

Authors

Luis Samaniego

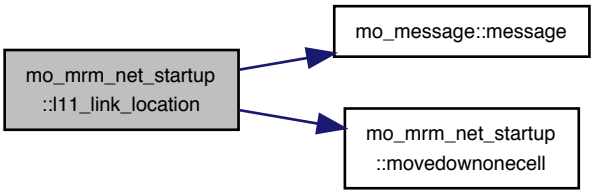
Date

Dec 2005

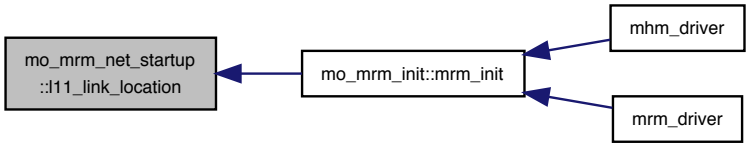
References mo_mrm_global_variables::basin_mrm, mo_common_variables::l0_basin, mo_mrm_global_variables::l0_drasc, mo_mrm_global_variables::l0_fdir, mo_mrm_global_variables::l11_colout, mo_mrm_global_variables::l11_fcol, mo_mrm_global_variables::l11_fromn, mo_mrm_global_variables::l11_frow, mo_mrm_global_variables::l11_netperm, mo_mrm_global_variables::l11_noutlets, mo_mrm_global_variables::l11_rowout, mo_mrm_global_variables::l11_tcol, mo_mrm_global_variables::l11_trow, mo_common_variables::level0, mo_mrm_global_variables::level11, mo_message::message(), movedownonecell(), and mo_common_constants::nodata_i4.

Referenced by mo_mrm_init::mrm_init().

Here is the call graph for this function:



Here is the caller graph for this function:



16.54.2.9 l11_routing_order()

```
subroutine, public mo_mrm_net_startup::l11_routing_order (
             integer(i4), intent(in) iBasin )
```

Find routing order, headwater cells and sink.

Find routing order, headwater cells and sink. If a variable is added or removed here, then it also has to be added or removed in the subroutine L11_config_set in module mo_restart and in the subroutine set_L11_config in module mo_set_netcdf_restart

Parameters

in	*integer(i4) :: iBasin*	Basin Id

Authors

Luis Samaniego

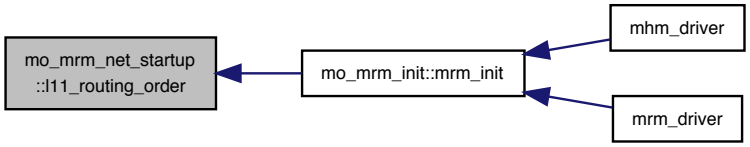
Date

Dec 2005

References mo_mrm_global_variables::l11_fdir, mo_mrm_global_variables::l11_fromn, mo_mrm_global_↩ variables::l11_label, mo_mrm_global_variables::l11_netperm, mo_mrm_global_variables::l11_noutlets, mo_↩ mrm_global_variables::l11_rorder, mo_mrm_global_variables::l11_sink, mo_mrm_global_variables::l11_ton, mo↩ _mrm_global_variables::level11, and mo_common_constants::nodata_i4.

Referenced by mo_mrm_init::mrm_init().

Here is the caller graph for this function:



16.54.2.10 l11_set_drain_outlet_gauges()

```
subroutine, public mo_mrm_net_startup::l11_set_drain_outlet_gauges (
             integer(i4), intent(in) iBasin )
```

Draining cell identification and Set gauging node.

Perform the following tasks:

- Draining cell identification (cell at L0 to draining cell outlet at L11).
- Set gauging nodes If a variable is added or removed here, then it also has to be added or removed in the subroutine L11_config_set in module mo_restart and in the subroutine set_L11_config in module mo_set_↩ netcdf_restart

Parameters

in	*integer(i4) :: iBasin*	Basin Id

Authors

Luis Samaniego

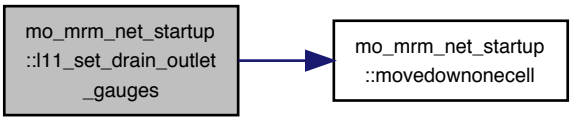
Date

Dec 2005

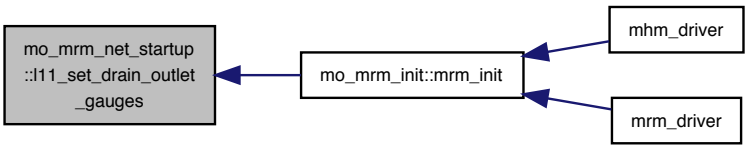
References mo_mrm_global_variables::basin_mrm, mo_common_variables::l0_basin, mo_mrm_global_↩ variables::l0_dracell, mo_mrm_global_variables::l0_drasc, mo_mrm_global_variables::l0_fdir, mo_mrm_global_↩ variables::l0_gaugeloc, mo_mrm_global_variables::l0_inflowgaugeloc, mo_mrm_global_variables::l0_l11_remap, mo_common_variables::level0, movedownonecell(), and mo_common_constants::nodata_i4.

Referenced by mo_mrm_init::mrm_init().

Here is the call graph for this function:



Here is the caller graph for this function:



16.54.2.11 l11_set_network_topology()

```
subroutine, public mo_mrm_net_startup::l11_set_network_topology (
             integer(i4), intent(in) iBasin )
```

Set network topology.

Set network topology from and to node for all links at level-11 (Routing Network). If a variable is added or removed here, then it also has to be added or removed in the subroutine L11_config_set in module mo_restart and in the subroutine set_L11_config in module mo_set_netcdf_restart.

Parameters

in	*integer(i4) :: iBasin*	Basin Id

Authors

Luis Samaniego

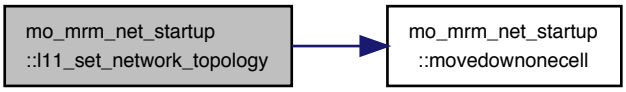
Date

Dec 2005

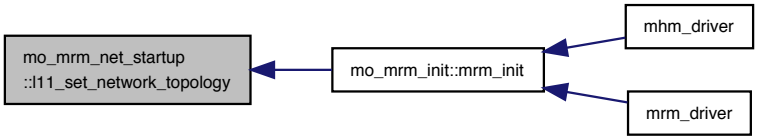
References mo_mrm_global_variables::l11_fdir, mo_mrm_global_variables::l11_fromn, mo_mrm_global_↩ variables::l11_ton, mo_mrm_global_variables::level11, movedownonecell(), and mo_common_constants::nodata↩ _i4.

Referenced by mo_mrm_init::mrm_init().

Here is the call graph for this function:



Here is the caller graph for this function:



16.54.2.12 l11_stream_features()

```
subroutine, public mo_mrm_net_startup::l11_stream_features (
            integer(i4), intent(in) iBasin )
```

Stream features (stream network and floodplain)

Stream features (stream network and floodplain) If a variable is added or removed here, then it also has to be added or removed in the subroutine L11_config_set in module mo_restart and in the subroutine set_L11_config in module mo_set_netcdf_restart

Parameters

in	*integer(i4) :: iBasin*	Basin Id

Authors

Luis Samaniego

Date

Dec 2005

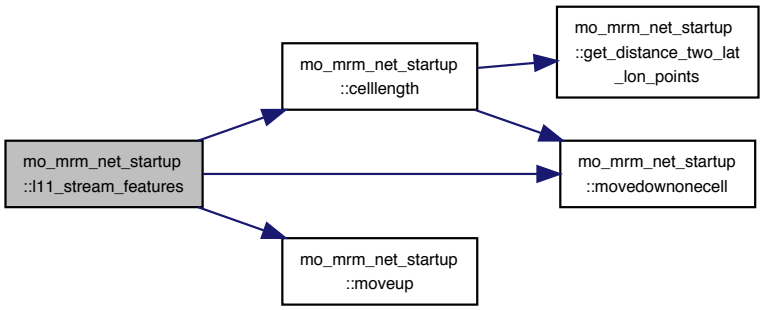
stack(nNodes, 2)

References celllength(), mo_common_variables::iflag_cordinate_sys, mo_common_variables::l0_basin, mo_↩ common_variables::l0_elev, mo_mrm_global_variables::l0_fdir, mo_mrm_global_variables::l0_floodplain, mo↩

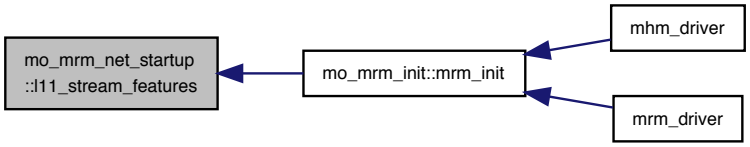
_mrm_global_variables::l0_streamnet, mo_mrm_global_variables::l11_afloodplain, mo_mrm_global_variables↩ ::l11_fcol, mo_mrm_global_variables::l11_frow, mo_mrm_global_variables::l11_length, mo_mrm_global_↩ variables::l11_netperm, mo_mrm_global_variables::l11_noutlets, mo_mrm_global_variables::l11_slope, mo_↩ mrm_global_variables::l11_tcol, mo_mrm_global_variables::l11_trow, mo_common_variables::level0, mo_mrm↩ _global_variables::level11, movedownonecell(), moveup(), mo_common_constants::nodata_dp, mo_common_↩ constants::nodata_i4, and mo_common_variables::processmatrix.

Referenced by mo_mrm_init::mrm_init().

Here is the call graph for this function:



Here is the caller graph for this function:



16.54.2.13 movedownonecell()

```
subroutine mo_mrm_net_startup::movedownonecell (
             integer(i4), intent(in) fDir,
             integer(i4), intent(inout) iRow,
             integer(i4), intent(inout) jCol )
```

TODO: add description.

TODO: add description

Parameters

in	*integer(i4) :: fDir*	
in,out	*integer(i4) :: iRow, jCol*	

Parameters

in,out	*integer(i4) :: iRow, jCol*	

Authors

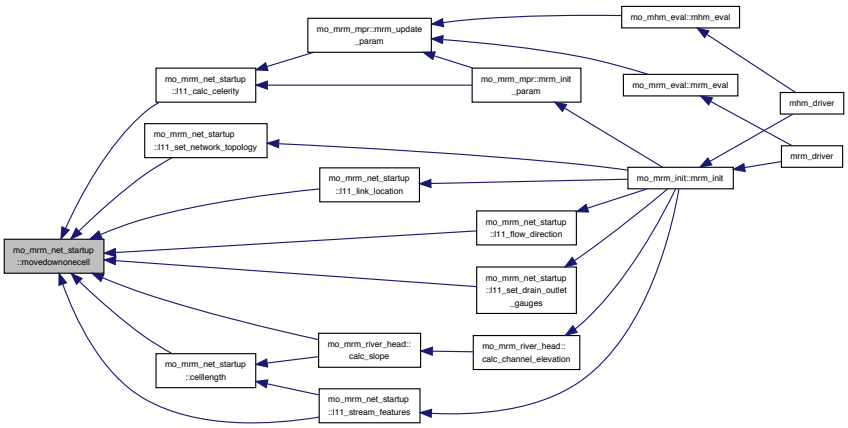
Robert Schweppe

Date

Jun 2018

Referenced by mo_mrm_river_head::calc_slope(), celllength(), l11_calc_celerity(), l11_flow_direction(), l11_link_↩ location(), l11_set_drain_outlet_gauges(), l11_set_network_topology(), and l11_stream_features().

Here is the caller graph for this function:



16.54.2.14 moveup()

```
subroutine mo_mrm_net_startup::moveup (
            real(dp), dimension(:, :), intent(in), allocatable elev0,
            integer(i4), dimension(:, :), intent(in), allocatable fDir0,
            integer(i4), intent(in) fi,
            integer(i4), intent(in) fj,
            integer(i4), dimension(:, :), intent(inout) ss,
            integer(i4), intent(inout) nn )
```

TODO: add description.

TODO: add description

Parameters

in	*real(dp), dimension(:, :) :: elev0*	
in	*integer(i4), dimension(:, :) :: fDir0*	
in	*integer(i4) :: fi, fj*	co-ordinate of the stream bed
in	*integer(i4) :: fi, fj*	co-ordinate of the stream bed
in,out	*integer(i4), dimension(:, :) :: ss*	
in,out	*integer(i4) :: nn*	

Authors

Robert Schweppe

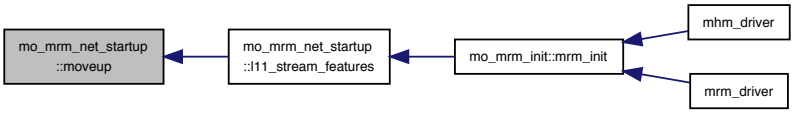
Date

Jun 2018

References mo_mrm_constants::deltah.

Referenced by l11_stream_features().

Here is the caller graph for this function:



16.55 mo_mrm_objective_function_runoff Module Reference

Objective Functions for Optimization of mHM/mRM against runoff.

Functions/Subroutines

- real(dp) function, public single_objective_runoff (parameterset, eval, arg1, arg2, arg3)

  Wrapper for objective functions optimizing agains runoff.
- subroutine, public multi_objective_runoff (parameterset, eval, multi_objectives)

  Wrapper for multi-objective functions where at least one is regarding runoff.
- real(dp) function loglikelihood_stddev (parameterset, eval, stddev, stddev_new, likeli_new)

  Logarithmic likelihood function with removed linear trend and Lag(1)-autocorrelation.
- real(dp) function loglikelihood_evin2013_2 (parameterset, eval, regularize)

  Logarithmised likelihood with linear error model and lag(1)-autocorrelation of the relative errors.
- real(dp) function parameter_regularization (paraset, prior, bounds, mask)

  TODO: add description.
- real(dp) function loglikelihood_trend_no_autocorr (parameterset, eval, stddev_old, stddev_new, likeli_new)

  Logarithmic likelihood function with linear trend removed.
- real(dp) function objective_lnnse (parameterset, eval)

  Objective function of logarithmic NSE.
- real(dp) function objective_sse (parameterset, eval)

  Objective function of SSE.
- real(dp) function objective_nse (parameterset, eval)

  Objective function of NSE.
- real(dp) function objective_equal_nse_lnnse (parameterset, eval)

  Objective function equally weighting NSE and lnNSE.
- real(dp) function, dimension(2) multi_objective_nse_lnnse (parameterset, eval)

  Multi-objective function with NSE and lnNSE.
- real(dp) function, dimension(2) multi_objective_lnnse_highflow_lnnse_lowflow (parameterset, eval)

  Multi-objective function with NSE and lnNSE.

- real(dp) function, dimension(2) multi_objective_lnnse_highflow_lnnse_lowflow_2 (parameterset, eval)

  Multi-objective function with NSE and lnNSE.
- real(dp) function, dimension(2) multi_objective_ae_fdc_lsv_nse_djf (parameterset, eval)

  Multi-objective function with absolute error of Flow Duration Curves low-segment volume and nse of DJF's discharge.
- real(dp) function objective_power6_nse_lnnse (parameterset, eval)

  Objective function of combined NSE and lnNSE with power of 5 i.e. the p-norm with p=5.
- real(dp) function objective_kge (parameterset, eval)

  Objective function of KGE.
- real(dp) function objective_multiple_gauges_kge_power6 (parameterset, eval)

  combined objective function based on KGE raised to the power 6
- real(dp) function objective_weighted_nse (parameterset, eval)

  Objective function of weighted NSE.
- subroutine, public extract_runoff (gaugeId, runoff, runoff_agg, runoff_obs, runoff_obs_mask)

  extracts runoff data from global variables

16.55.1 Detailed Description

Objective Functions for Optimization of mHM/mRM against runoff.

This module provides a wrapper for several objective functions used to optimize mRM/mHM against runoff. If the objective contains besides runoff another variable like TWS move it to mHM/mo_objective_function.f90. If it is only regarding runoff implement it here. All the objective functions are supposed to be minimized! (1) SO: Q: 1.0 - NSE (2) SO: Q: 1.0 - lnNSE (3) SO: Q: 1.0 - 0.5∗(NSE+lnNSE) (4) SO: Q: -1.0 ∗ loglikelihood with trend removed from absolute errors and then lag(1)-autocorrelation removed (5) SO: Q: ((1-NSE)∗∗6+(1-lnNSE)∗∗6)∗∗(1/6) (6) SO: Q: SSE (7) SO: Q: -1.0 ∗ loglikelihood with trend removed from absolute errors (8) SO: Q: -1.0 ∗ loglikelihood with trend removed from the relative errors and then lag(1)-autocorrelation removed (9) SO: Q: 1.0 - KGE (Kling-Gupta efficiency measure) (14) SO: Q: sum[((1.0-KGE_i)/ nGauges)∗∗6]∗∗(1/6) > combination of KGE of every gauging station based on a power-6 norm (16) MO: Q: 1st objective: 1.0 - NSE Q: 2nd objective: 1.0 - lnNSE (18) MO: Q: 1st objective: 1.0 - lnNSE(Q_highflow) (95% percentile) Q: 2nd objective: 1.0 - lnNSE(Q_lowflow) (5% of data range) (19) MO: Q: 1st objective: 1.0 - lnNSE(Q_highflow) (non-low flow) Q: 2nd objective: 1.0 - lnNSE(Q_lowflow) (5% of data range)eshold for Q (20) MO: Q: 1st objective: absolute difference in FDC's low-segment volume Q: 2nd objective: 1.0 - NSE of discharge of months DJF (31) SO: Q: 1.0 - wNSE - weighted NSE

Authors

Juliane Mai

Date

Dec 2012

16.55.2 Function/Subroutine Documentation

16.55.2.1 extract_runoff()

```
subroutine, public mo_mrm_objective_function_runoff::extract_runoff (
            integer(i4), intent(in) gaugeId,
            real(dp), dimension(:, :), intent(in) runoff,
            real(dp), dimension(:), intent(out), allocatable runoff_agg,
            real(dp), dimension(:), intent(out), allocatable runoff_obs,
            logical, dimension(:), intent(out), allocatable runoff_obs_mask )
```

extracts runoff data from global variables

extracts simulated and measured runoff from global variables, such that they overlay exactly. For measured runoff, only the runoff during the evaluation period are cut, not succeeding nodata values. For simulated runoff, warming days as well as succeeding nodata values are neglected and the simulated runoff is aggregated to the resolution of the observed runoff. see use in this module above

Parameters

in	*integer(i4) :: gaugeId*	current gauge Id
in	*real(dp), dimension(:, :) :: runoff*	simulated runoff
out	*real(dp), dimension(:) :: runoff_agg*	aggregated simulated
out	*real(dp), dimension(:) :: runoff_obs*	extracted measured
out	*logical, dimension(:) :: runoff_obs_mask*	mask of no data values

Authors

Stephan Thober

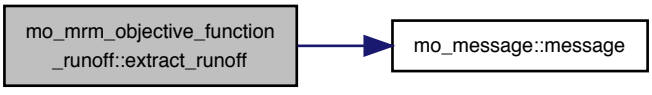
Date

Jan 2015

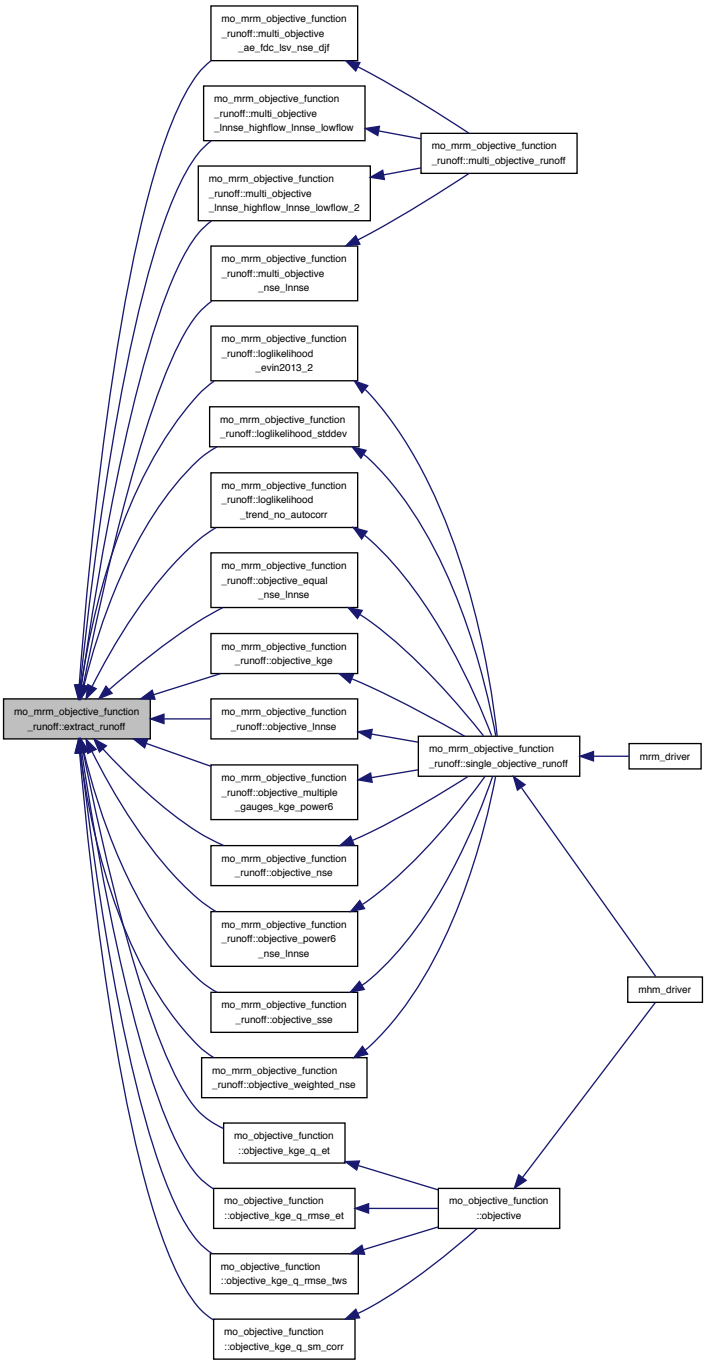
References mo_common_mhm_mrm_variables::evalper, mo_mrm_global_variables::gauge, mo_message::message(), mo_mrm_global_variables::nmeasperday, mo_common_mhm_mrm_variables::ntstepday, and mo_common_mhm_mrm_variables::warmingdays.

Referenced by loglikelihood_evin2013_2(), loglikelihood_stddev(), loglikelihood_trend_no_autocorr(), multi_objective_ae_fdc_lsv_nse_djf(), multi_objective_lnnse_highflow_lnnse_lowflow(), multi_objective_lnnse_highflow_lnnse_lowflow_2(), multi_objective_nse_lnnse(), objective_equal_nse_lnnse(), objective_kge(), mo_objective_function::objective_kge_q_et(), mo_objective_function::objective_kge_q_rmse_et(), mo_objective_function::objective_kge_q_rmse_tws(), mo_objective_function::objective_kge_q_sm_corr(), objective_lnnse(), objective_multiple_gauges_kge_power6(), objective_nse(), objective_power6_nse_lnnse(), objective_sse(), and objective_weighted_nse().

Here is the call graph for this function:



Here is the caller graph for this function:



16.55.2.2 loglikelihood_evin2013_2()

```
real(dp) function mo_mrm_objective_function_runoff::loglikelihood_evin2013_2 (
            real(dp), dimension(:), intent(in) parameterset,
            procedure(eval_interface), intent(in), pointer eval,
```

```
            logical, intent(in), optional regularize )
```

Logarithmised likelihood with linear error model and lag(1)-autocorrelation of the relative errors.

This loglikelihood uses a linear error model and a lag(1)-autocorrelation on the relative errors. This is approach 2 of the paper Evin et al. (WRR, 2013). This is opti_function = 8. mHM then adds two extra (local) parameters for the error model in mhm_driver, which get optimised together with the other, global parameters. ADDITIONAL INFORMATION Evin et al., WRR 49, 4518-4524, 2013

Parameters

in	*real(dp), dimension(:) :: parameterset*	
in	*procedure(eval_interface) :: eval*	
in	*logical, optional :: regularize*	

Returns

real(dp) :: loglikelihood_evin2013_2 — logarithmic likelihood using given stddev but remove optimal trend and lag(1)-autocorrelation in errors (absolute between running model with parameterset and observation)

Authors

Juliane Mai and Matthias Cuntz

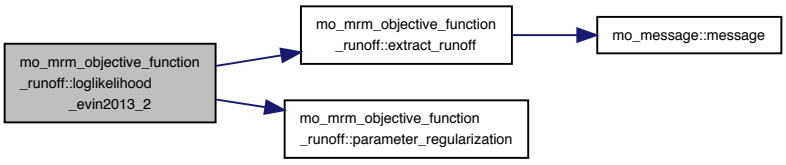
Date

Mar 2014

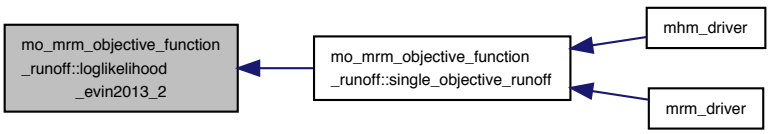
References extract_runoff(), mo_common_variables::global_parameters, parameter_regularization(), and mo_↩ constants::pi_dp.

Referenced by single_objective_runoff().

Here is the call graph for this function:



Here is the caller graph for this function:



16.55.2.3 loglikelihood_stddev()

```
real(dp) function mo_mrm_objective_function_runoff::loglikelihood_stddev (
             real(dp), dimension(:), intent(in) parameterset,
             procedure(eval_interface), intent(in), pointer eval,
             real(dp), intent(in) stddev,
             real(dp), intent(out), optional stddev_new,
             real(dp), intent(out), optional likeli_new )
```

Logarithmic likelihood function with removed linear trend and Lag(1)-autocorrelation.

The logarithmis likelihood function is used when mHM runs in MCMC mode. It can also be used for optimization when selecting the likelihood in the namelist as *opti_function*.

Parameters

in	*real(dp), dimension(:) :: parameterset*	
in	*procedure(eval_interface) :: eval*	
in	*real(dp) :: stddev*	standard deviation of data
out	*real(dp), optional :: stddev_new*	standard deviation of errors with removed trend and correlationbetween model run using parameter set and observation
out	*real(dp), optional :: likeli_new*	logarithmic likelihood determined with stddev_new instead of stddev

Returns

real(dp) :: loglikelihood_stddev — logarithmic likelihood using given stddev but remove optimal trend and lag(1)-autocorrelation in errors (absolute between running model with parameterset and observation)

Authors

Juliane Mai

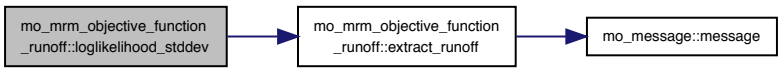
Date

Dec 2012

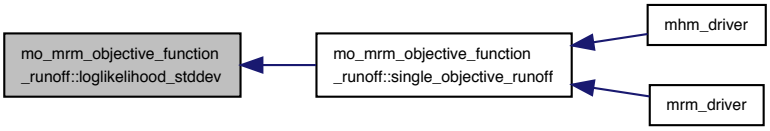
References extract_runoff().

Referenced by single_objective_runoff().

Here is the call graph for this function:



Here is the caller graph for this function:



16.55.2.4 loglikelihood_trend_no_autocorr()

```
real(dp) function mo_mrm_objective_function_runoff::loglikelihood_trend_no_autocorr (
             real(dp), dimension(:), intent(in) parameterset,
             procedure(eval_interface), intent(in), pointer eval,
             real(dp), intent(in) stddev_old,
             real(dp), intent(out), optional stddev_new,
             real(dp), intent(out), optional likeli_new )
```

Logarithmic likelihood function with linear trend removed.

The logarithmis likelihood function is used when mHM runs in MCMC mode. It can also be used for optimization when selecting the likelihood in the namelist as *opti_function*.

Parameters

in	*real(dp), dimension(:) :: parameterset*	
in	*procedure(eval_interface) :: eval*	
in	*real(dp) :: stddev_old*	standard deviation of data
out	*real(dp), optional :: stddev_new*	standard deviation of errors with removed trendbetween model run using parameter set and observation
out	*real(dp), optional :: likeli_new*	logarithmic likelihood determined with stddev_new instead of stddev

Returns

real(dp) :: loglikelihood_trend_no_autocorr — logarithmic likelihood using given stddev but remove optimal trend in errors (absolute between running model with parameterset and observation)

Authors

Juliane Mai and Matthias Cuntz

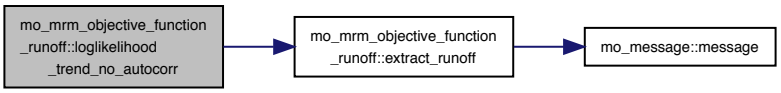
Date

Mar 2014

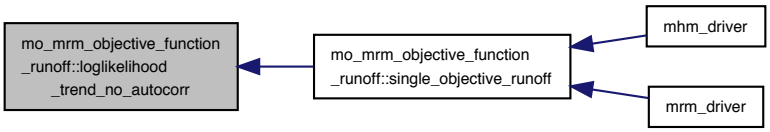
References extract_runoff().

Referenced by single_objective_runoff().

Here is the call graph for this function:



Here is the caller graph for this function:



16.55.2.5 multi_objective_ae_fdc_lsv_nse_djf()

```
real(dp) function, dimension(2) mo_mrm_objective_function_runoff::multi_objective_ae_fdc_lsv_↩
nse_djf (
             real(dp), dimension(:), intent(in) parameterset,
             procedure(eval_interface), intent(in), pointer eval )
```

Multi-objective function with absolute error of Flow Duration Curves low-segment volume and nse of DJF's discharge.

The objective function only depends on a parameter vector. The model will be called with that parameter vector and the model output is subsequently compared to observed data. The first objective is using the routine "↩ FlowDurationCurves" from "mo_signatures" to determine the low-segment volume of the FDC. The objective is the absolute difference between the observed volume and the simulated volume. For the second objective the discharge of the winter months December, January and February are extracted from the time series. The objective is then the Nash-Sutcliffe efficiency NSE of the observed winter discharge against the simulated winter discharge. The observed data Q_{obs} are global in this module. To calibrate this objective you need a multi-objective optimizer like PA-DDS.

Parameters

in	*real(dp), dimension(:) :: parameterset*	
in	*procedure(eval_interface) :: eval*	

Returns

real(dp), dimension(2) :: multi_objective_ae_fdc_lsv_nse_djf — objective function value (which will be e.g. minimized by an optimization routine like PA-DDS)

Authors

Juliane Mai

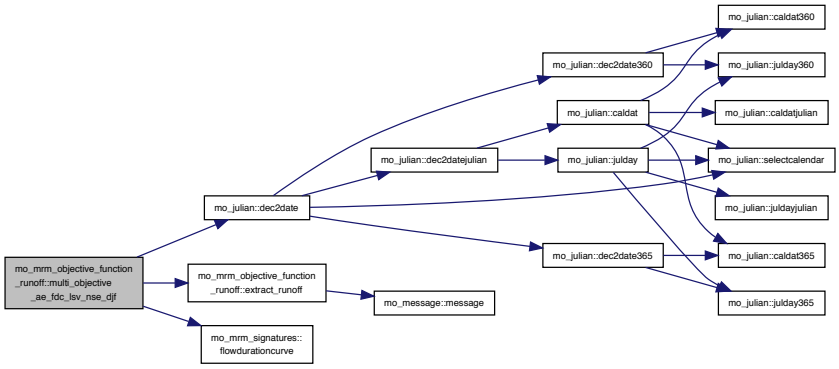
Date

Feb 2016

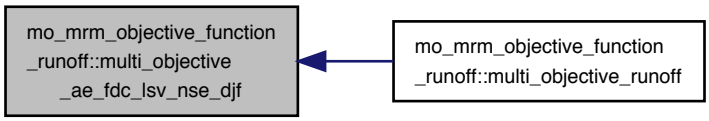
References mo_julian::dec2date(), mo_common_mhm_mrm_variables::evalper, extract_runoff(), mo_mrm_↩ signatures::flowdurationcurve(), mo_mrm_global_variables::gauge, and mo_mrm_global_variables::nmeasperday.

Referenced by multi_objective_runoff().

Here is the call graph for this function:



Here is the caller graph for this function:



16.55.2.6 multi_objective_lnnse_highflow_lnnse_lowflow()

```
real(dp) function, dimension(2) mo_mrm_objective_function_runoff::multi_objective_lnnse_↩
highflow_lnnse_lowflow (
            real(dp), dimension(:), intent(in) parameterset,
            procedure(eval_interface), intent(in), pointer eval )
```

Multi-objective function with NSE and lnNSE.

The objective function only depends on a parameter vector. The model will be called with that parameter vector and the model output is subsequently compared to observed data. A timepoint t of the observed data is marked as a lowflow timepoint t_{low} if

$$Q_{obs}(t) < min(Q_{obs}) + 0.05 * (max(Q_{obs}) - min(Q_{obs}))$$

and a timepoint t of the observed data is marked as a highflow timepoint t_{high} if

$$t_{high} if Q_{obs}(i) > percentile(Q_{obs}, 95.)$$

This timepoint identification is only performed for the observed data. The first objective is the logarithmic Nash-↩ Sutcliffe model efficiency coefficient $lnNSE_{high}$ of discharge values at high-flow timepoints

$$lnNSE_{high} = 1 - \frac{\sum_{i=1}^{N_{high}} (\ln Q_{obs}(i) - \ln Q_{model}(i))^2}{\sum_{i=1}^{N} (\ln Q_{obs}(i) - \ln \bar{Q}_{obs})^2}$$

. The second objective is the logarithmic Nash-Sutcliffe model efficiency coefficient $lnNSE_{low}$ of discharge values at low-flow timepoints

$$lnNSE_{low} = 1 - \frac{\sum_{i=1}^{N_{low}} (\ln Q_{obs}(i) - \ln Q_{model}(i))^2}{\sum_{i=1}^{N} (\ln Q_{obs}(i) - \ln \bar{Q}_{obs})^2}$$

. Both objectives are returned. The observed data Q_{obs} are global in this module. To calibrate this objective you need a multi-objective optimizer like PA-DDS.

Parameters

in	*real(dp), dimension(:) :: parameterset*	
in	*procedure(eval_interface) :: eval*	

Returns

real(dp), dimension(2) :: multi_objective_lnnse_highflow_lnnse_lowflow — objective function value (which will be e.g. minimized by an optimization routine like PA-DDS)

Authors

Juliane Mai

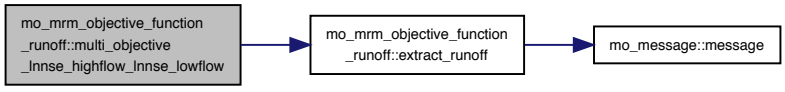
Date

Oct 2015

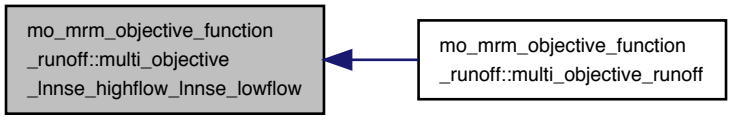
References extract_runoff().

Referenced by multi_objective_runoff().

Here is the call graph for this function:



Here is the caller graph for this function:



16.55.2.7 multi_objective_lnnse_highflow_lnnse_lowflow_2()

```
real(dp) function, dimension(2) mo_mrm_objective_function_runoff::multi_objective_lnnse_↩
highflow_lnnse_lowflow_2 (
            real(dp), dimension(:), intent(in) parameterset,
            procedure(eval_interface), intent(in), pointer eval )
```

Multi-objective function with NSE and lnNSE.

The objective function only depends on a parameter vector. The model will be called with that parameter vector and the model output is subsequently compared to observed data. A timepoint t of the observed data is marked as a lowflow timepoint t_{low} if

$$Q_{obs}(t) < min(Q_{obs}) + 0.05 * (max(Q_{obs}) - min(Q_{obs}))$$

and all other timepoints are marked as a highflow timepoints t_{high}. This timepoint identification is only performed for the observed data. The first objective is the logarithmic Nash-Sutcliffe model efficiency coefficient $lnNSE_{high}$ of discharge values at high-flow timepoints

$$lnNSE_{high} = 1 - \frac{\sum_{i=1}^{N_{high}} (\ln Q_{obs}(i) - \ln Q_{model}(i))^2}{\sum_{i=1}^{N} (\ln Q_{obs}(i) - \ln \bar{Q}_{obs})^2}$$

. The second objective is the logarithmic Nash-Sutcliffe model efficiency coefficient $lnNSE_{low}$ of discharge values at low-flow timepoints

$$lnNSE_{low} = 1 - \frac{\sum_{i=1}^{N_{low}} (\ln Q_{obs}(i) - \ln Q_{model}(i))^2}{\sum_{i=1}^{N} (\ln Q_{obs}(i) - \ln \bar{Q}_{obs})^2}$$

. Both objectives are returned. The observed data Q_{obs} are global in this module. To calibrate this objective you need a multi-objective optimizer like PA-DDS.

Parameters

in	*real(dp), dimension(:) :: parameterset*	
in	*procedure(eval_interface) :: eval*	

Returns

real(dp), dimension(2) :: multi_objective_lnnse_highflow_lnnse_lowflow_2 — objective function value (which will be e.g. minimized by an optimization routine like PA-DDS)

Authors

Juliane Mai

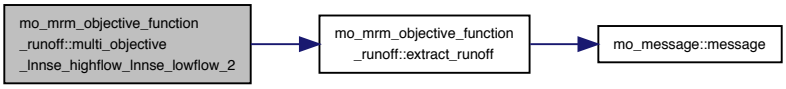
Date

Oct 2015

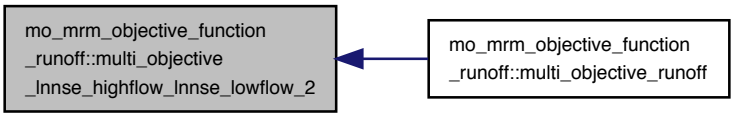
References extract_runoff().

Referenced by multi_objective_runoff().

Here is the call graph for this function:



Here is the caller graph for this function:



16.55.2.8 multi_objective_nse_lnnse()

```
real(dp) function, dimension(2) mo_mrm_objective_function_runoff::multi_objective_nse_lnnse (
            real(dp), dimension(:), intent(in) parameterset,
            procedure(eval_interface), intent(in), pointer eval )
```

Multi-objective function with NSE and lnNSE.

The objective function only depends on a parameter vector. The model will be called with that parameter vector and the model output is subsequently compared to observed data. Therefore, the Nash-Sutcliffe model efficiency coefficient NSE

$$NSE = 1 - \frac{\sum_{i=1}^{N}(Q_{obs}(i) - Q_{model}(i))^2}{\sum_{i=1}^{N}(Q_{obs}(i) - \bar{Q_{obs}})^2}$$

and the logarithmic Nash-Sutcliffe model efficiency coefficient $lnNSE$

$$lnNSE = 1 - \frac{\sum_{i=1}^{N}(\ln Q_{obs}(i) - \ln Q_{model}(i))^2}{\sum_{i=1}^{N}(\ln Q_{obs}(i) - \ln \bar{Q}_{obs})^2}$$

are calculated and both returned. The observed data Q_{obs} are global in this module. To calibrate this objective you need a multi-objective optimizer like PA-DDS.

Parameters

in	*real(dp), dimension(:) :: parameterset*	
in	*procedure(eval_interface) :: eval*	

Returns

real(dp), dimension(2) :: multi_objective_nse_lnnse — objective function value (which will be e.g. minimized by an optimization routine like PA-DDS)

Authors

Juliane Mai

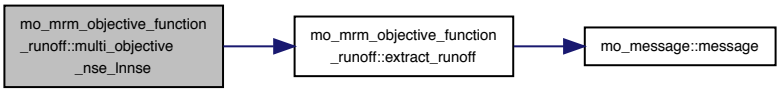
Date

Oct 2015

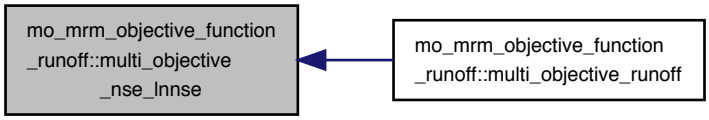
References extract_runoff().

Referenced by multi_objective_runoff().

Here is the call graph for this function:



Here is the caller graph for this function:



16.55.2.9 multi_objective_runoff()

```
subroutine, public mo_mrm_objective_function_runoff::multi_objective_runoff (
             real(dp), dimension(:), intent(in) parameterset,
             procedure(eval_interface), intent(in), pointer eval,
             real(dp), dimension(:), intent(out), allocatable multi_objectives )
```

Wrapper for multi-objective functions where at least one is regarding runoff.

The functions selects the objective function case defined in a namelist, i.e. the global variable *opti_function*. It return the multiple objective function values for a specific parameter set.

Parameters

in	*REAL(dp), DIMENSION(:) :: parameterset*	
in	*procedure(eval_interface) :: eval*	
out	*REAL(dp), DIMENSION(:) :: multi_objectives*	

Authors

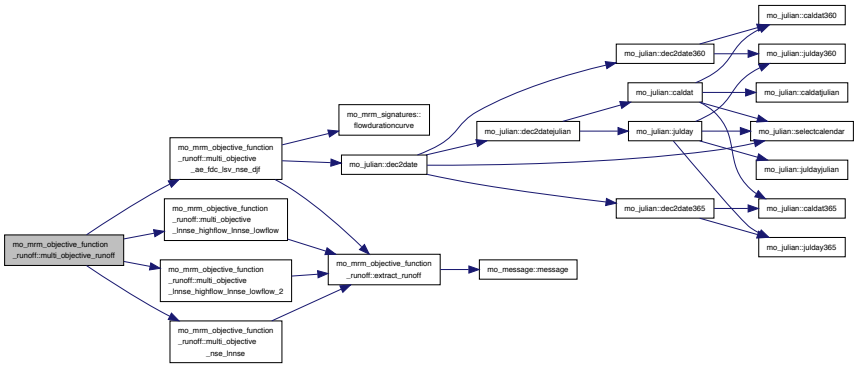
Juliane Mai

Date

Oct 2015

References multi_objective_ae_fdc_lsv_nse_djf(), multi_objective_lnnse_highflow_lnnse_lowflow(), multi_↩ objective_lnnse_highflow_lnnse_lowflow_2(), multi_objective_nse_lnnse(), and mo_common_mhm_mrm_↩ variables::opti_function.

Here is the call graph for this function:



16.55.2.10 objective_equal_nse_lnnse()

```
real(dp) function mo_mrm_objective_function_runoff::objective_equal_nse_lnnse (
            real(dp), dimension(:), intent(in) parameterset,
            procedure(eval_interface), intent(in), pointer eval )
```

Objective function equally weighting NSE and lnNSE.

The objective function only depends on a parameter vector. The model will be called with that parameter vector and the model output is subsequently compared to observed data. Therefore, the Nash-Sutcliffe model efficiency coefficient NSE

$$NSE = 1 - \frac{\sum_{i=1}^{N}(Q_{obs}(i) - Q_{model}(i))^2}{\sum_{i=1}^{N}(Q_{obs}(i) - \bar{Q_{obs}})^2}$$

and the logarithmic Nash-Sutcliffe model efficiency coefficient $lnNSE$

$$lnNSE = 1 - \frac{\sum_{i=1}^{N}(\ln Q_{obs}(i) - \ln Q_{model}(i))^2}{\sum_{i=1}^{N}(\ln Q_{obs}(i) - \ln \bar{Q}_{obs})^2}$$

are calculated and added up equally weighted:

$$obj\_value = \frac{1}{2}(NSE + lnNSE)$$

The observed data Q_{obs} are global in this module.

Parameters

in	*real(dp), dimension(:) :: parameterset*	
in	*procedure(eval_interface) :: eval*	

Returns

real(dp) :: objective_equal_nse_lnnse — objective function value (which will be e.g. minimized by an optimization routine like DDS)

Authors

Juliane Mai

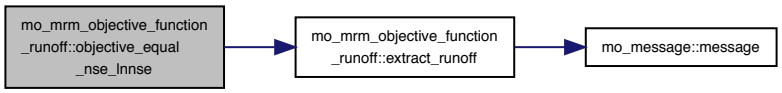
Date

May 2013

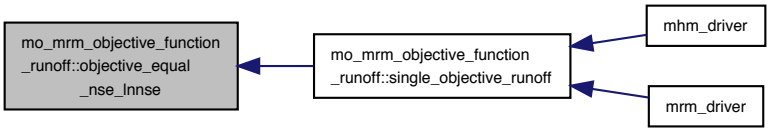
References extract_runoff().

Referenced by single_objective_runoff().

Here is the call graph for this function:



Here is the caller graph for this function:



16.55.2.11 objective_kge()

```
real(dp) function mo_mrm_objective_function_runoff::objective_kge (
            real(dp), dimension(:), intent(in) parameterset,
            procedure(eval_interface), intent(in), pointer eval )
```

Objective function of KGE.

The objective function only depends on a parameter vector. The model will be called with that parameter vector and the model output is subsequently compared to observed data. Therefore, the Kling-Gupta model efficiency coefficient KGE

$$KGE = 1.0 - \sqrt{((1-r)^2 + (1-\alpha)^2 + (1-\beta)^2)}$$

where r = Pearson product-moment correlation coefficient α = ratio of similated mean to observed mean β = ratio of similated standard deviation to observed standard deviation is calculated and the objective function is

$$obj\_value = 1.0 - KGE$$

$(1-KGE)$ is the objective since we always apply minimization methods. The minimal value of $(1-KGE)$ is 0 for the optimal KGE of 1.0. The observed data Q_{obs} are global in this module.

Parameters

in	*real(dp), dimension(:) :: parameterset*	
in	*procedure(eval_interface) :: eval*	

Returns

real(dp) :: objective_kge — objective function value (which will be e.g. minimized by an optimization routine like DDS)

Authors

Rohini Kumar

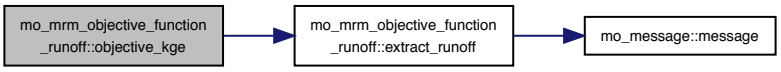
Date

August 2014

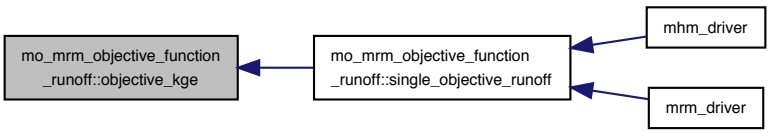
References extract_runoff().

Referenced by single_objective_runoff().

Here is the call graph for this function:



Here is the caller graph for this function:



16.55.2.12 objective_lnnse()

```
real(dp) function mo_mrm_objective_function_runoff::objective_lnnse (
            real(dp), dimension(:), intent(in) parameterset,
            procedure(eval_interface), intent(in), pointer eval )
```

Objective function of logarithmic NSE.

The objective function only depends on a parameter vector. The model will be called with that parameter vector and the model output is subsequently compared to observed data. Therefore, the logarithmic Nash-Sutcliffe model efficiency coefficient $lnNSE$

$$lnNSE = 1 - \frac{\sum_{i=1}^{N}(\ln Q_{obs}(i) - \ln Q_{model}(i))^2}{\sum_{i=1}^{N}(\ln Q_{obs}(i) - \ln \bar{Q}_{obs})^2}$$

is calculated.

$$obj\_value = lnNSE$$

The observed data Q_{obs} are global in this module.

Parameters

in	*real(dp), dimension(:) :: parameterset*	
in	*procedure(eval_interface) :: eval*	

Returns

real(dp) :: objective_lnnse — objective function value (which will be e.g. minimized by an optimization routine like DDS)

Authors

Juliane Mai

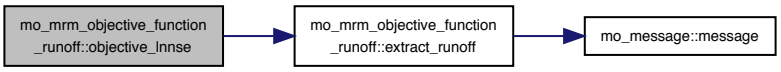
Date

May 2013

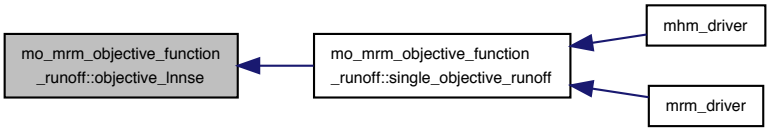
References extract_runoff().

Referenced by single_objective_runoff().

Here is the call graph for this function:



Here is the caller graph for this function:



16.55.2.13 objective_multiple_gauges_kge_power6()

```
real(dp) function mo_mrm_objective_function_runoff::objective_multiple_gauges_kge_power6 (
            real(dp), dimension(:), intent(in) parameterset,
            procedure(eval_interface), intent(in), pointer eval )
```

combined objective function based on KGE raised to the power 6

The objective function only depends on a parameter vector. The model will be called with that parameter vector and the model output is subsequently compared to observed data. Therefore, the Kling-Gupta model efficiency coefficient KGE for a given gauging station

$$KGE = 1.0 - \sqrt{((1-r)^2 + (1-\alpha)^2 + (1-\beta)^2)}$$

where r = Pearson product-moment correlation coefficient α = ratio of similated mean to observed mean β = ratio of similated standard deviation to observed standard deviation is calculated and the objective function for a given gauging station (i) is

$$\phi_i = 1.0 - KGE_i$$

ϕ_i is the objective since we always apply minimization methods. The minimal value of ϕ_i is 0 for the optimal KGE of 1.0. Finally, the overall OF is estimated based on the power-6 norm to combine the ϕ_i from all gauging stations (N).

$$OF = \sqrt[6]{\sum((1.0 - KGE_i)/N)^6}$$

. The observed data Q_{obs} are global in this module.

Parameters

in	*real(dp), dimension(:) :: parameterset*	
in	*procedure(eval_interface) :: eval*	

Returns

real(dp) :: objective_multiple_gauges_kge_power6 — objective function value (which will be e.g. minimized by an optimization routine like DDS)

Authors

Rohini Kumar

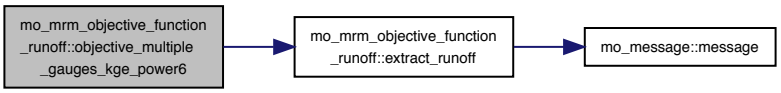
Date

March 2015

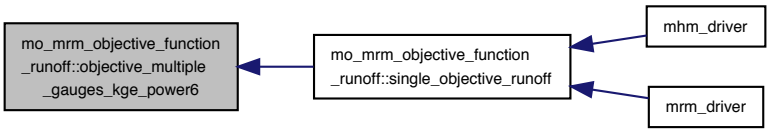
References extract_runoff().

Referenced by single_objective_runoff().

Here is the call graph for this function:



Here is the caller graph for this function:



16.55.2.14 objective_nse()

```
real(dp) function mo_mrm_objective_function_runoff::objective_nse (
            real(dp), dimension(:), intent(in) parameterset,
            procedure(eval_interface), intent(in), pointer eval )
```

Objective function of NSE.

The objective function only depends on a parameter vector. The model will be called with that parameter vector and the model output is subsequently compared to observed data. Therefore, the Nash-Sutcliffe model efficiency coefficient NSE

$$NSE = 1 - \frac{\sum_{i=1}^{N}(Q_{obs}(i) - Q_{model}(i))^2}{\sum_{i=1}^{N}(Q_{obs}(i) - \bar{Q_{obs}})^2}$$

is calculated and the objective function is

$$obj\_value = 1 - NSE$$

The observed data Q_{obs} are global in this module.

Parameters

in	*real(dp), dimension(:) :: parameterset*	
in	*procedure(eval_interface) :: eval*	

Returns

real(dp) :: objective_nse — objective function value (which will be e.g. minimized by an optimization routine like DDS)

Authors

Juliane Mai

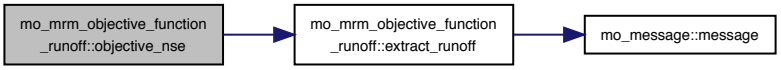
Date

May 2013

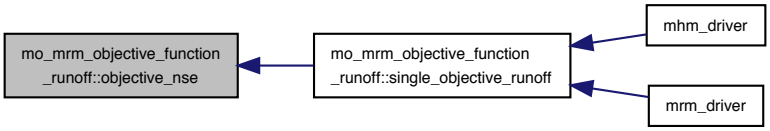
References extract_runoff().

Referenced by single_objective_runoff().

Here is the call graph for this function:



Here is the caller graph for this function:



16.55.2.15 objective_power6_nse_lnnse()

```
real(dp) function mo_mrm_objective_function_runoff::objective_power6_nse_lnnse (
            real(dp), dimension(:), intent(in) parameterset,
            procedure(eval_interface), intent(in), pointer eval )
```

Objective function of combined NSE and lnNSE with power of 5 i.e. the p-norm with p=5.

The objective function only depends on a parameter vector. The model will be called with that parameter vector and the model output is subsequently compared to observed data. Therefore, the Nash-Sutcliffe model efficiency coefficient NSE

$$NSE = 1 - \frac{\sum_{i=1}^{N}(Q_{obs}(i) - Q_{model}(i))^2}{\sum_{i=1}^{N}(Q_{obs}(i) - \bar{Q_{obs}})^2}$$

and the logarithmic Nash-Sutcliffe model efficiency coefficient $lnNSE$

$$lnNSE = 1 - \frac{\sum_{i=1}^{N}(\ln Q_{obs}(i) - \ln Q_{model}(i))^2}{\sum_{i=1}^{N}(\ln Q_{obs}(i) - \ln \bar{Q}_{obs})^2}$$

are calculated and added up equally weighted:

$$obj\_value = \sqrt[6]{(1-NSE)^6 + (1-lnNSE)^6}$$

The observed data Q_{obs} are global in this module.

Parameters

in	*real(dp), dimension(:) :: parameterset*	
in	*procedure(eval_interface) :: eval*	

Returns

real(dp) :: objective_power6_nse_lnnse — objective function value (which will be e.g. minimized by an optimization routine like DDS)

Authors

Juliane Mai and Matthias Cuntz

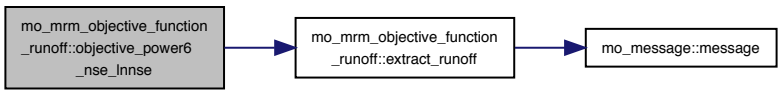
Date

March 2014

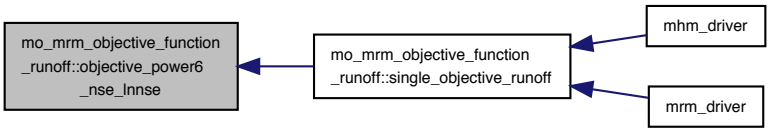
References extract_runoff().

Referenced by single_objective_runoff().

Here is the call graph for this function:



Here is the caller graph for this function:



16.55.2.16 objective_sse()

```
real(dp) function mo_mrm_objective_function_runoff::objective_sse (
            real(dp), dimension(:), intent(in) parameterset,
            procedure(eval_interface), intent(in), pointer eval )
```

Objective function of SSE.

The objective function only depends on a parameter vector. The model will be called with that parameter vector and the model output is subsequently compared to observed data. Therefore, the sum squared errors

$$SSE = \sum_{i=1}^{N} (Q_{obs}(i) - Q_{model}(i))^2$$

is calculated and the objective function is

$$obj\_value = SSE$$

The observed data Q_{obs} are global in this module.

Parameters

in	*real(dp), dimension(:) :: parameterset*	
in	*procedure(eval_interface) :: eval*	

Returns

real(dp) :: objective_sse — objective function value (which will be e.g. minimized by an optimization routine like DDS)

Authors

Juliane Mai and Matthias Cuntz

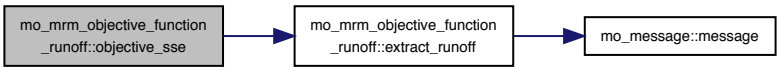
Date

March 2014

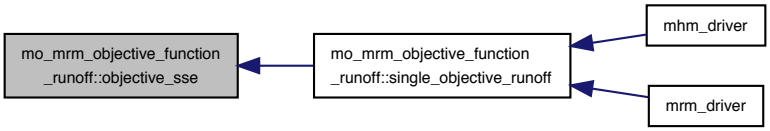
References extract_runoff().

Referenced by single_objective_runoff().

Here is the call graph for this function:



Here is the caller graph for this function:



16.55.2.17 objective_weighted_nse()

```
real(dp) function mo_mrm_objective_function_runoff::objective_weighted_nse (
             real(dp), dimension(:), intent(in) parameterset,
             procedure(eval_interface), intent(in), pointer eval )
```

Objective function of weighted NSE.

The objective function only depends on a parameter vector. The model will be called with that parameter vector and the model output is subsequently compared to observed data. Therefore, the weighted Nash-Sutcliffe model efficiency coefficient NSE

$$wNSE = 1 - \frac{\sum_{i=1}^{N} Q_{obs}(i) * (Q_{obs}(i) - Q_{model}(i))^2}{\sum_{i=1}^{N} Q_{obs}(i) * (Q_{obs}(i) - \bar{Q_{obs}})^2}$$

is calculated and the objective function is

$$obj\_value = 1 - wNSE$$

The observed data Q_{obs} are global in this module.

Parameters

in	*real(dp), dimension(:) :: parameterset*	
in	*procedure(eval_interface) :: eval*	

Returns

real(dp) :: objective_weighted_nse — objective function value (which will be e.g. minimized by an optimization routine like DDS)

Authors

Stephan Thober, Bjoern Guse

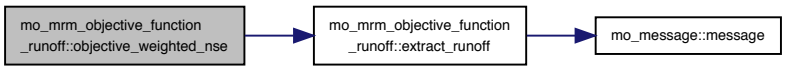
Date

May 2018

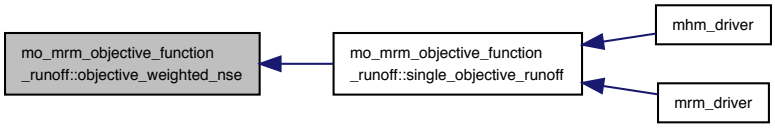
References extract_runoff().

Referenced by single_objective_runoff().

Here is the call graph for this function:



Here is the caller graph for this function:



16.55.2.18 parameter_regularization()

```
real(dp) function mo_mrm_objective_function_runoff::parameter_regularization (
             real(dp), dimension(:), intent(in) paraset,
             real(dp), dimension(size(paraset)), intent(in) prior,
             real(dp), dimension(size(paraset), 2), intent(in) bounds,
             logical, dimension(size(paraset)), intent(in) mask )
```

TODO: add description.

TODO: add description

Parameters

in	*real(dp), dimension(:) :: paraset*	
in	*real(dp), dimension(size(paraset)) :: prior*	
in	*real(dp), dimension(size(paraset), 2) :: bounds*	(min, max)
in	*logical, dimension(size(paraset)) :: mask*	

Authors

Robert Schweppe

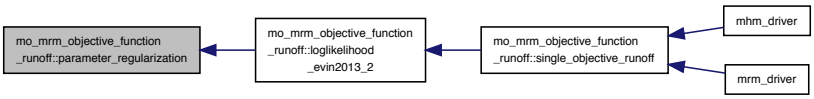
Date

Jun 2018

References mo_constants::pi_dp.

Referenced by loglikelihood_evin2013_2().

Here is the caller graph for this function:



16.55.2.19 single_objective_runoff()

```
real(dp) function, public mo_mrm_objective_function_runoff::single_objective_runoff (
            real(dp), dimension(:), intent(in) parameterset,
            procedure(eval_interface), intent(in), pointer eval,
            real(dp), intent(in), optional arg1,
            real(dp), intent(out), optional arg2,
            real(dp), intent(out), optional arg3 )
```

Wrapper for objective functions optimizing agains runoff.

The functions selects the objective function case defined in a namelist, i.e. the global variable *opti_function*. It return the objective function value for a specific parameter set.

Parameters

in	*REAL(dp), DIMENSION(:) :: parameterset*	
in	*procedure(eval_interface) :: eval*	
in	*real(dp), optional :: arg1*	
out	*real(dp), optional :: arg2*	
out	*real(dp), optional :: arg3*	

Returns

real(dp) :: objective — objective function value (which will be e.g. minimized by an optimization routine like DDS)

Authors

Juliane Mai

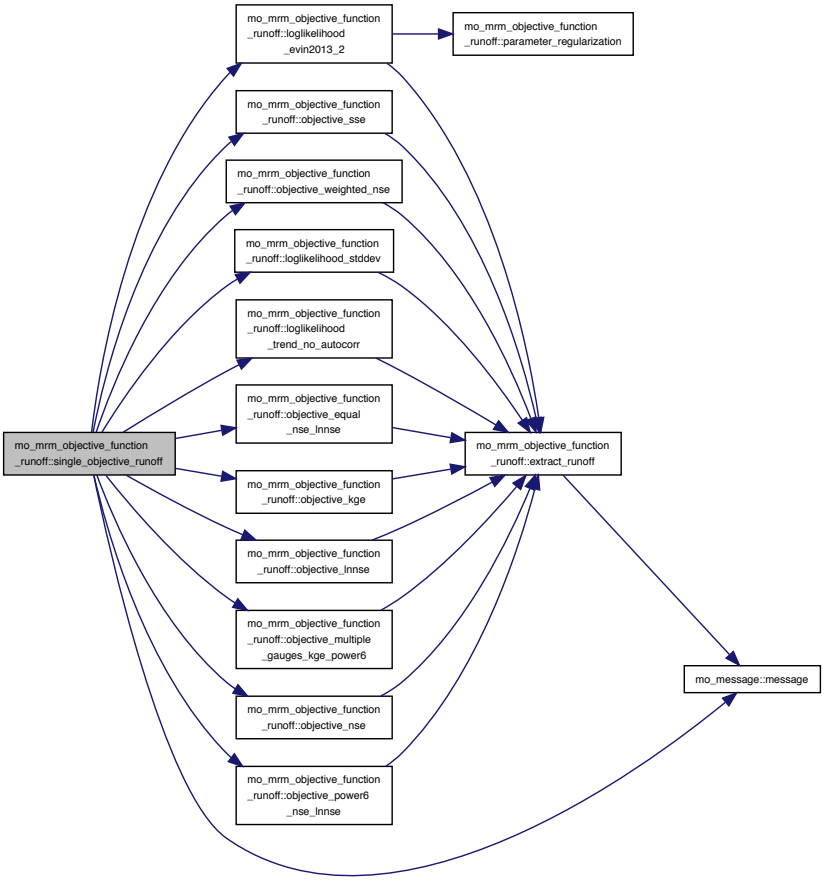
Date

Dec 2012

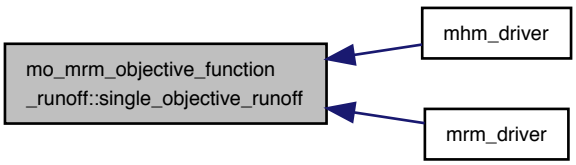
References loglikelihood_evin2013_2(), loglikelihood_stddev(), loglikelihood_trend_no_autocorr(), mo_message↩ ::message(), objective_equal_nse_lnnse(), objective_kge(), objective_lnnse(), objective_multiple_gauges_kge↩ _power6(), objective_nse(), objective_power6_nse_lnnse(), objective_sse(), objective_weighted_nse(), mo_↩ common_mhm_mrm_variables::opti_function, and mo_common_mhm_mrm_variables::opti_method.

Referenced by mhm_driver(), and mrm_driver().

Here is the call graph for this function:



Here is the caller graph for this function:



16.56 mo_mrm_read_config Module Reference

read mRM config

Functions/Subroutines

- subroutine, public mrm_read_config (file_namelist, unamelist, file_namelist_param, unamelist_param, do_↩ message, readLatLon)

  Read the general config of mRM.
- subroutine read_mrm_routing_params (processCase, file_namelist_param, unamelist_param)

  TODO: add description.

16.56.1 Detailed Description

read mRM config

This module contains all mRM subroutines related to reading the mRM configuration either from file or copy from mHM.

Authors

Stephan Thober

Date

Aug 2015

16.56.2 Function/Subroutine Documentation

16.56.2.1 mrm_read_config()

```
subroutine, public mo_mrm_read_config::mrm_read_config (
             character(*), intent(in) file_namelist,
             integer, intent(in) unamelist,
             character(*), intent(in) file_namelist_param,
             integer, intent(in) unamelist_param,
             logical, intent(in) do_message,
             logical, intent(out) readLatLon )
```

Read the general config of mRM.

Depending on the variable mrm_coupling_config, the mRM config is either read from mrm.nml and parameters from mrm_parameter.nml or copied from mHM.

Parameters

in	*character(∗) :: file_namelist, file_namelist_param*	
in	*integer :: unamelist, unamelist_param*	
in	*character(∗) :: file_namelist, file_namelist_param*	
in	*integer :: unamelist, unamelist_param*	
in	*logical :: do_message*	- flag for writing mHM standard messages
out	*logical :: readLatLon*	- flag for reading LatLon file

Authors

Stephan Thober

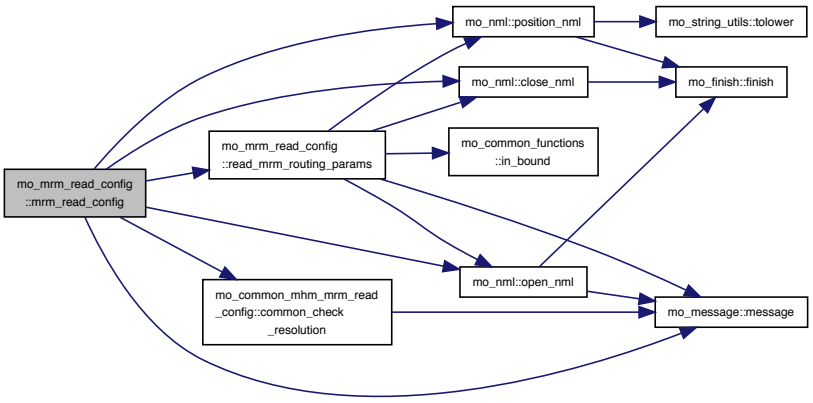
Date

Aug 2015

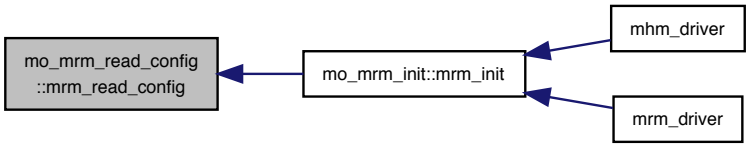
References mo_common_variables::alma_convention, mo_mrm_global_variables::basin_mrm, mo_nml↩ ::close_nml(), mo_common_mhm_mrm_read_config::common_check_resolution(), mo_mrm_global_variables↩ ::dirbankfullrunoff, mo_mrm_global_variables::dirgauges, mo_mrm_global_variables::dirtotalrunoff, mo_mrm↩ _file::file_defoutput, mo_mrm_global_variables::filenametotalrunoff, mo_mrm_global_variables::gauge, mo_↩ mrm_global_variables::gw_coupling, mo_mrm_global_variables::inflowgauge, mo_mrm_global_variables::is_start, mo_common_constants::maxnobasins, mo_mrm_constants::maxnogauges, mo_message::message(), mo_↩ common_variables::nbasins, mo_mrm_global_variables::ngaugestotal, mo_mrm_global_variables::ninflowgaugestotal, mo_common_constants::nodata_i4, mo_nml::open_nml(), mo_mrm_global_variables::outputflxstate_mrm, mo_nml::position_nml(), mo_common_variables::processmatrix, read_mrm_routing_params(), mo_mrm_↩ global_variables::timestep_model_outputs_mrm, mo_mrm_file::udefoutput, and mo_mrm_global_variables↩ ::varnametotalrunoff.

Referenced by mo_mrm_init::mrm_init().

Here is the call graph for this function:



Here is the caller graph for this function:



16.56.2.2 read_mrm_routing_params()

```
subroutine mo_mrm_read_config::read_mrm_routing_params (
            integer(i4), intent(in) processCase,
```

```
character(*), intent(in) file_namelist_param,
integer(i4), intent(in) unamelist_param )
```

TODO: add description.

TODO: add description

Parameters

in	*integer(i4) :: processCase*	it is the default case, should be one
in	*character(*) :: file_namelist_param*	file name containing parameter namelist
in	*integer(i4) :: unamelist_param*	file name id containing parameter namelist

Authors

Robert Schweppe

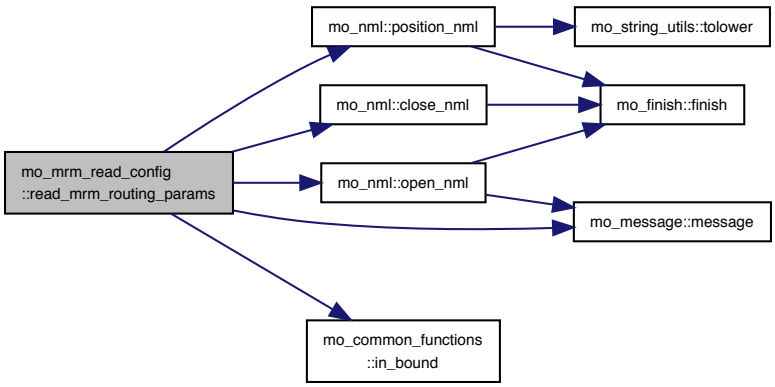
Date

Jun 2018

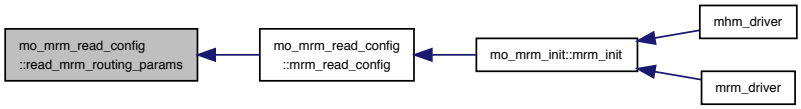
References mo_nml::close_nml(), mo_common_variables::global_parameters, mo_common_variables::global↩ _parameters_name, mo_common_functions::in_bound(), mo_message::message(), mo_common_constants↩ ::ncolpars, mo_nml::open_nml(), mo_nml::position_nml(), and mo_common_variables::processmatrix.

Referenced by mrm_read_config().

Here is the call graph for this function:



Here is the caller graph for this function:



16.57 mo_mrm_read_data Module Reference

This module contains all routines to read mRM data from file.

Functions/Subroutines

- subroutine, public mrm_read_l0_data (do_reinit, do_readlatlon, do_readlcover)

  read L0 data from file
- subroutine, public mrm_read_discharge

  Read discharge timeseries from file.
- subroutine, public mrm_read_total_runoff (iBasin)

  read simulated runoff that is to be routed
- subroutine, public mrm_read_bankfull_runoff (iBasin)
- subroutine rotate_fdir_variable (x)

  TODO: add description.

16.57.1 Detailed Description

This module contains all routines to read mRM data from file.

TODO: add description

Authors

Stephan Thober

Date

Aug 2015

16.57.2 Function/Subroutine Documentation

16.57.2.1 mrm_read_bankfull_runoff()

```
subroutine, public mo_mrm_read_data::mrm_read_bankfull_runoff (
             integer(i4), intent(in) iBasin )
```

Parameters

in	*ibasin*	

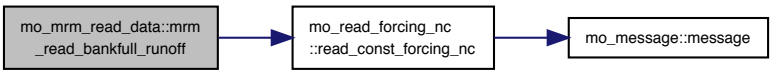
Note

The file read in must contain a double precision float variable with the name "Q_bkfl".

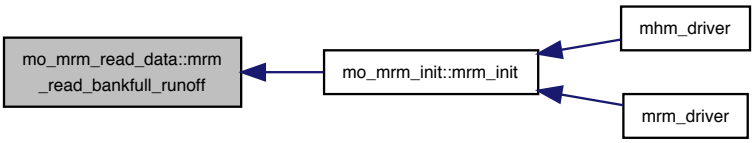
References mo_common_variables::alma_convention, mo_mrm_global_variables::dirbankfullrunoff, mo_common_constants::hoursecs, mo_mrm_global_variables::level11, mo_read_forcing_nc::read_const_forcing_nc(), and mo_common_mhm_mrm_variables::timestep.

Referenced by mo_mrm_init::mrm_init().

Here is the call graph for this function:



Here is the caller graph for this function:



16.57.2.2 mrm_read_discharge()

```
subroutine, public mo_mrm_read_data::mrm_read_discharge ( )
```

Read discharge timeseries from file.

Read Observed discharge at the outlet of a catchment and at the inflow of a catchment. Allocate global runoff variable that contains the simulated runoff after the simulation.

Authors

Matthias Zink & Stephan Thober

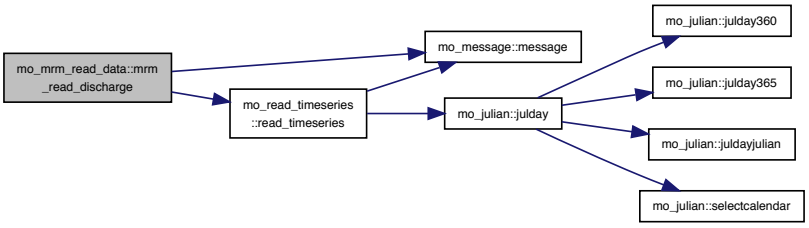
Date

Aug 2015

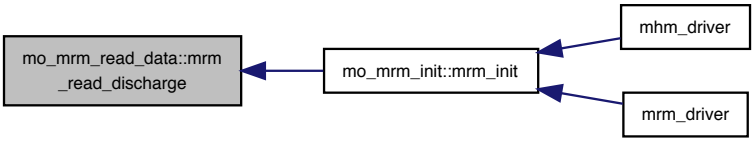
References mo_common_mhm_mrm_variables::evalper, mo_mrm_global_variables::gauge, mo_mrm_global↩ _variables::inflowgauge, mo_message::message(), mo_mrm_global_variables::mrm_runoff, mo_common_↩ variables::nbasins, mo_mrm_global_variables::ngaugestotal, mo_mrm_global_variables::ninflowgaugestotal, mo↩ _mrm_global_variables::nmeasperday, mo_common_constants::nodata_dp, mo_common_mhm_mrm_variables↩ ::ntstepday, mo_common_mhm_mrm_variables::opti_function, mo_common_mhm_mrm_variables::optimize, mo↩ _read_timeseries::read_timeseries(), mo_common_mhm_mrm_variables::simper, and mo_mrm_file::udischarge.

Referenced by mo_mrm_init::mrm_init().

Here is the call graph for this function:



Here is the caller graph for this function:



16.57.2.3 mrm_read_l0_data()

```
subroutine, public mo_mrm_read_data::mrm_read_l0_data (
             logical, intent(in) do_reinit,
             logical, intent(in) do_readlatlon,
             logical, intent(in) do_readlcover )
```

read L0 data from file

With the exception of L0_mask, L0_elev, and L0_LCover, all L0 variables are read from file. The former three are only read if they are not provided as variables.

Parameters

in	*logical :: do_reinit*	
in	*logical :: do_readlatlon*	
in	*logical :: do_readlcover*	

Authors

Juliane Mai, Matthias Zink, and Stephan Thober

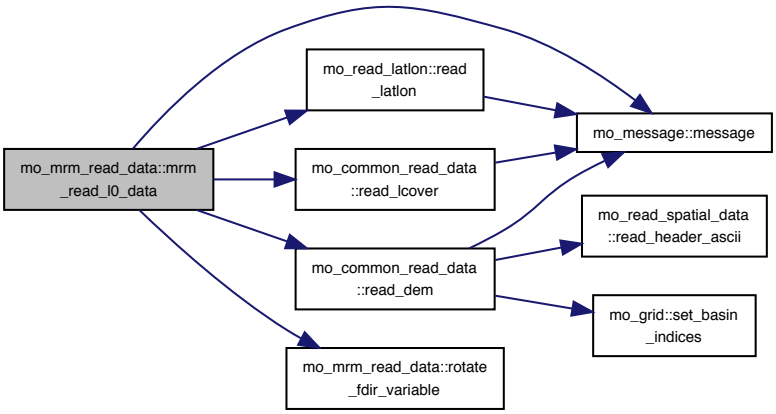
Date

Aug 2015

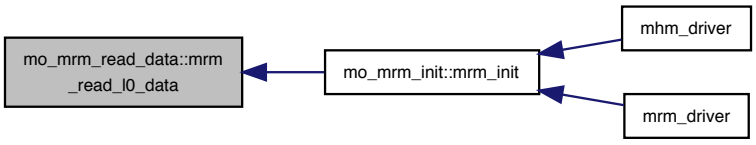
References mo_mrm_global_variables::basin_mrm, mo_common_variables::dirmorpho, mo_mrm_file::file_facc, mo_mrm_file::file_fdir, mo_mrm_file::file_gaugeloc, mo_mpr_file::file_slope, mo_common_variables::l0_basin, mo_mrm_global_variables::l0_facc, mo_mrm_global_variables::l0_fdir, mo_mrm_global_variables::l0_gaugeloc, mo_mrm_global_variables::l0_inflowgaugeloc, mo_common_variables::l0_lcover, mo_mpr_global_variables::l0_slope, mo_common_variables::level0, mo_message::message(), mo_common_variables::nbasins, mo_common_constants::nodata_dp, mo_common_constants::nodata_i4, mo_common_variables::processmatrix, mo_common_read_data::read_dem(), mo_read_latlon::read_latlon(), mo_common_read_data::read_lcover(), rotate_fdir_variable(), mo_mrm_file::ufacc, mo_mrm_file::ufdir, mo_mrm_file::ugaugeloc, and mo_mpr_file::uslope.

Referenced by mo_mrm_init::mrm_init().

Here is the call graph for this function:



Here is the caller graph for this function:



16.57.2.4 mrm_read_total_runoff()

```
subroutine, public mo_mrm_read_data::mrm_read_total_runoff (
            integer(i4), intent(in) iBasin )
```

read simulated runoff that is to be routed

read spatio-temporal field of total runoff that has been simulated by a hydrologic model or land surface model. This total runoff will then be aggregated to the level 11 resolution and then routed through the stream network.

Parameters

in	*integer(i4) :: iBasin*	basin id

Authors

Stephan Thober

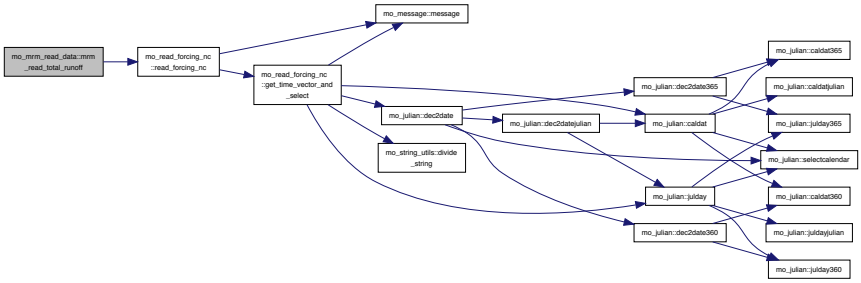
Date

Sep 2015

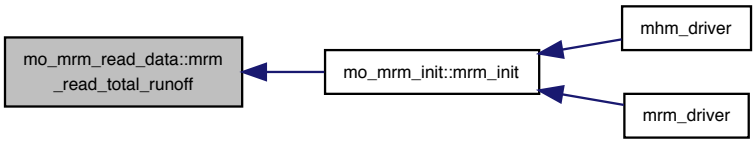
References mo_common_variables::alma_convention, mo_mrm_global_variables::dirtotalrunoff, mo_mrm_↩ global_variables::filenametotalrunoff, mo_common_constants::hoursecs, mo_mrm_global_variables::l1_total↩ _runoff_in, mo_common_variables::level1, mo_common_constants::nodata_dp, mo_read_forcing_nc::read_↩ forcing_nc(), mo_common_mhm_mrm_variables::simper, mo_common_mhm_mrm_variables::timestep, and mo↩ _mrm_global_variables::varnametotalrunoff.

Referenced by mo_mrm_init::mrm_init().

Here is the call graph for this function:



Here is the caller graph for this function:



16.57.2.5 rotate_fdir_variable()

```
subroutine mo_mrm_read_data::rotate_fdir_variable (
            integer(i4), dimension(:, :), intent(inout) x )
```

TODO: add description.

TODO: add description

Parameters

in,out	*integer(i4), dimension(:, :) :: x*	

Authors

L. Samaniego & R. Kumar

Date

Jun 2018

References mo_common_constants::nodata_i4.

Referenced by mrm_read_l0_data().

Here is the caller graph for this function:

16.58 mo_mrm_restart Module Reference

Restart routines.

Functions/Subroutines

- subroutine, public mrm_write_restart (iBasin, OutPath)
  write routing states and configuration
- subroutine, public mrm_read_restart_states (iBasin, InPath)
  read routing states
- subroutine, public mrm_read_restart_config (iBasin, InPath)
  reads Level 11 configuration from a restart directory

16.58.1 Detailed Description

Restart routines.

This module contains the subroutines for reading and writing routing related variables to file.

Authors

Stephan Thober

Date

Aug 2015

16.58.2 Function/Subroutine Documentation

16.58.2.1 mrm_read_restart_config()

```
subroutine, public mo_mrm_restart::mrm_read_restart_config (
             integer(i4), intent(in) iBasin,
             character(256), intent(in) InPath )
```

reads Level 11 configuration from a restart directory

read Level 11 configuration variables from a given restart directory and initializes all Level 11 configuration variables, that are initialized in L11_variable_init, contained in module mo_startup.

Parameters

in	*integer(i4) :: iBasin*	number of basin
in	*character(256) :: InPath*	Input Path including trailing slash

Authors

Stephan Thober

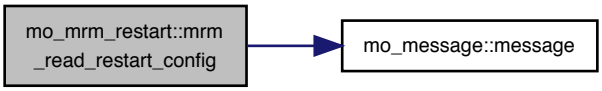
Date

Apr 2013

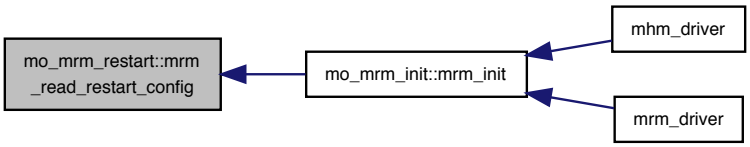
References mo_mrm_global_variables::basin_mrm, mo_kind::dp, mo_kind::i4, mo_common_variables::l0_basin, mo_mrm_global_variables::l0_facc, mo_mrm_global_variables::l0_fdir, mo_mpr_global_variables::l0_slope, mo_mrm_global_variables::l0_streamnet, mo_mrm_global_variables::l11_afloodplain, mo_mrm_global_variables::l11_colout, mo_mrm_global_variables::l11_facc, mo_mrm_global_variables::l11_fcol, mo_mrm_global_variables::l11_fdir, mo_mrm_global_variables::l11_fromn, mo_mrm_global_variables::l11_frow, mo_mrm_global_variables::l11_l1_id, mo_mrm_global_variables::l11_label, mo_mrm_global_variables::l11_length, mo_mrm_global_variables::l11_netperm, mo_mrm_global_variables::l11_noutlets, mo_mrm_global_variables::l11_rorder, mo_mrm_global_variables::l11_rowout, mo_mrm_global_variables::l11_sink, mo_mrm_global_variables::l11_slope, mo_mrm_global_variables::l11_tcol, mo_mrm_global_variables::l11_ton, mo_mrm_global_variables::l11_trow, mo_mrm_global_variables::l11_tsrout, mo_mrm_global_variables::l1_l11_id, mo_common_variables::level0, mo_common_variables::level1, mo_mrm_global_variables::level11, mo_message::message(), mo_common_variables::nbasins, mo_common_constants::nodata_dp, and mo_common_variables::processmatrix.

Referenced by mo_mrm_init::mrm_init().

Here is the call graph for this function:



Here is the caller graph for this function:



16.58.2.2 mrm_read_restart_states()

```
subroutine, public mo_mrm_restart::mrm_read_restart_states (
            integer(i4), intent(in) iBasin,
            character(256), intent(in) InPath )
```

read routing states

This subroutine reads the routing states from mRM_states_<basin_id>.nc that has to be in the given path directory. This subroutine has to be called directly each time the mHM_eval or mRM_eval is called such that the the states are always the same at the first simulation time step, crucial for optimization.

Parameters

in	*integer(i4) :: iBasin*	number of basin
in	*character(256) :: InPath*	Input Path including trailing slash

Authors

Stephan Thober

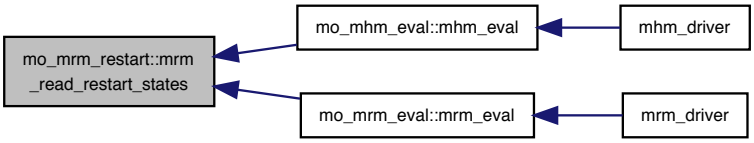
Date

Sep 2015

References mo_mrm_global_variables::l11_c1, mo_mrm_global_variables::l11_c2, mo_mrm_global_variables↩ ::l11_k, mo_mrm_global_variables::l11_nlinkfracfpimp, mo_mrm_global_variables::l11_qmod, mo_mrm_global↩ _variables::l11_qout, mo_mrm_global_variables::l11_qtin, mo_mrm_global_variables::l11_qtr, mo_mrm_global↩ _variables::l11_xi, mo_mrm_global_variables::level11, mo_common_variables::nlcoverscene, and mo_mrm_↩ constants::nroutingstates.

Referenced by mo_mhm_eval::mhm_eval(), and mo_mrm_eval::mrm_eval().

Here is the caller graph for this function:



16.58.2.3 mrm_write_restart()

```
subroutine, public mo_mrm_restart::mrm_write_restart (
             integer(i4), intent(in) iBasin,
             character(256), dimension(:), intent(in) OutPath )
```

write routing states and configuration

write configuration and state variables to a given restart directory.

Parameters

in	*integer(i4) :: iBasin*	number of basin
in	*character(256), dimension(:) :: OutPath*	list of Output paths per Basin

Authors

Stephan Thober

Date

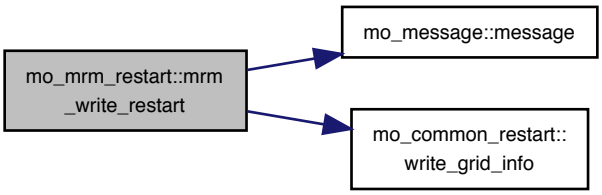
Aug 2015

References mo_mrm_global_variables::basin_mrm, mo_common_variables::l0_basin, mo_mrm_global_↩ variables::l0_facc, mo_mrm_global_variables::l0_fdir, mo_mpr_global_variables::l0_slope, mo_mrm_global_↩ variables::l0_streamnet, mo_mrm_global_variables::l11_afloodplain, mo_mrm_global_variables::l11_c1, mo_↩ mrm_global_variables::l11_c2, mo_mrm_global_variables::l11_celerity, mo_mrm_global_variables::l11_colout, mo_mrm_global_variables::l11_facc, mo_mrm_global_variables::l11_fcol, mo_mrm_global_variables::l11_fdir, mo_mrm_global_variables::l11_fromn, mo_mrm_global_variables::l11_frow, mo_mrm_global_variables::l11_↩ k, mo_mrm_global_variables::l11_l1_id, mo_mrm_global_variables::l11_label, mo_mrm_global_variables::l11↩

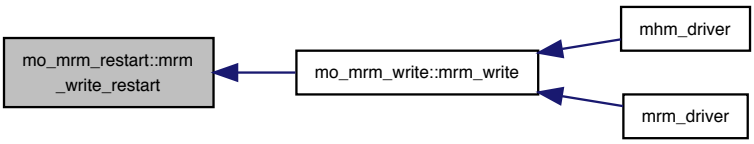
_length, mo_mrm_global_variables::l11_netperm, mo_mrm_global_variables::l11_nlinkfracfpimp, mo_mrm_↩ global_variables::l11_qmod, mo_mrm_global_variables::l11_qout, mo_mrm_global_variables::l11_qtin, mo_↩ mrm_global_variables::l11_qtr, mo_mrm_global_variables::l11_rorder, mo_mrm_global_variables::l11_rowout, mo_mrm_global_variables::l11_sink, mo_mrm_global_variables::l11_slope, mo_mrm_global_variables::l11_↩ tcol, mo_mrm_global_variables::l11_ton, mo_mrm_global_variables::l11_trow, mo_mrm_global_variables::l11↩ _tsrout, mo_mrm_global_variables::l11_xi, mo_mrm_global_variables::l1_l11_id, mo_common_variables::l1_↩ l11_remap, mo_common_variables::level0, mo_common_variables::level1, mo_mrm_global_variables::level11, mo_message::message(), mo_common_mhm_mrm_variables::mrm_coupling_mode, mo_common_variables↩ ::nlcoverscene, mo_common_constants::nodata_dp, mo_common_constants::nodata_i4, mo_mrm_constants↩ ::nroutingstates, mo_common_variables::processmatrix, and mo_common_restart::write_grid_info().

Referenced by mo_mrm_write::mrm_write().

Here is the call graph for this function:



Here is the caller graph for this function:



16.59 mo_mrm_river_head Module Reference

Functions/Subroutines

- subroutine, public init_masked_zeros_l0 (iBasin, data)
- subroutine, private reset_sum (iBasin, data)
- subroutine, public calc_channel_elevation ()
- subroutine, public calc_river_head (iBasin, L11_Qmod, river_head)
- real(dp) function calc_slope (iBasin, elev0, fDir0, i, j)
- subroutine, public avg_and_write_timestep (iBasin, timestep, data)
- subroutine, public create_output (iBasin, OutPath)

Variables

- type(ncvariable), dimension(:), allocatable nc_time
- type(ncvariable), dimension(:), allocatable nc_riverhead
- integer(i4), dimension(:), allocatable time_counter
- integer(i4), dimension(:), allocatable sum_counter

16.59.1 Function/Subroutine Documentation

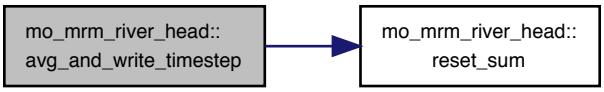
16.59.1.1 avg_and_write_timestep()

```
subroutine, public mo_mrm_river_head::avg_and_write_timestep (
             integer(i4), intent(in) iBasin,
             integer(i4), intent(in) timestep,
             real(dp), dimension(:), intent(inout) data )
```

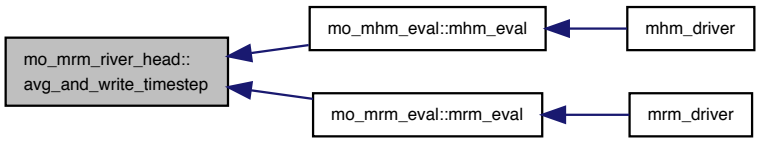
References mo_common_variables::level0, nc_riverhead, nc_time, mo_common_constants::nodata_dp, reset_sum(), sum_counter, and time_counter.

Referenced by mo_mhm_eval::mhm_eval(), and mo_mrm_eval::mrm_eval().

Here is the call graph for this function:



Here is the caller graph for this function:



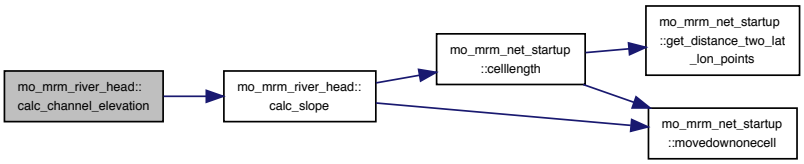
16.59.1.2 calc_channel_elevation()

```
subroutine, public mo_mrm_river_head::calc_channel_elevation ( )
```

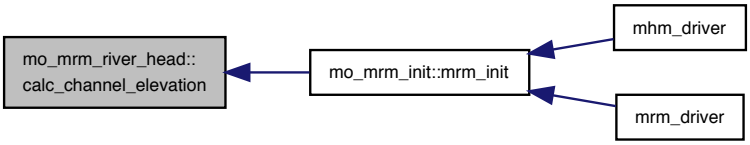
References calc_slope(), mo_mrm_global_variables::l0_channel_depth, mo_common_variables::l0_elev, mo_↩ mrm_global_variables::l0_facc, mo_mrm_global_variables::l0_fdir, mo_common_variables::nbasins, and mo_↩ common_constants::nodata_i4.

Referenced by mo_mrm_init::mrm_init().

Here is the call graph for this function:



Here is the caller graph for this function:



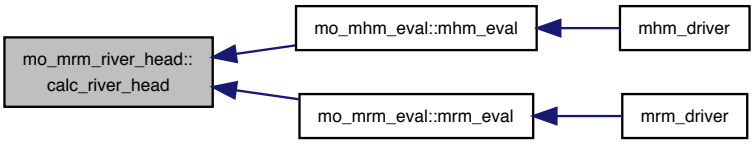
16.59.1.3 calc_river_head()

```
subroutine, public mo_mrm_river_head::calc_river_head (
             integer(i4), intent(in) iBasin,
             real(dp), dimension(:), intent(in) L11_Qmod,
             real(dp), dimension(:), intent(inout), allocatable river_head )
```

References mo_mrm_global_variables::l0_channel_elevation, mo_mrm_global_variables::l0_l11_remap, mo↩ _mrm_global_variables::l0_slope, mo_mrm_global_variables::l11_bankfull_runoff_in, mo_common_variables↩ ::level0, and sum_counter.

Referenced by mo_mhm_eval::mhm_eval(), and mo_mrm_eval::mrm_eval().

Here is the caller graph for this function:



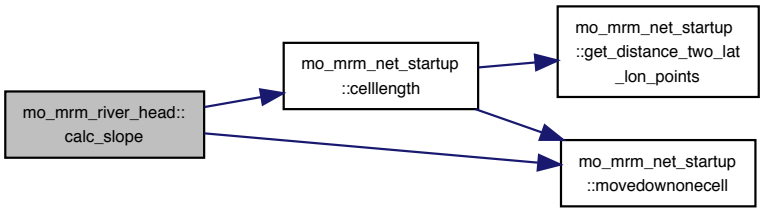
16.59.1.4 calc_slope()

```
real(dp) function mo_mrm_river_head::calc_slope (
            integer(i4), intent(in) iBasin,
            real(dp), dimension(:,:), intent(in), allocatable elev0,
            integer(i4), dimension(:,:), intent(in), allocatable fDir0,
            integer(i4), intent(in) i,
            integer(i4), intent(in) j )  [private]
```

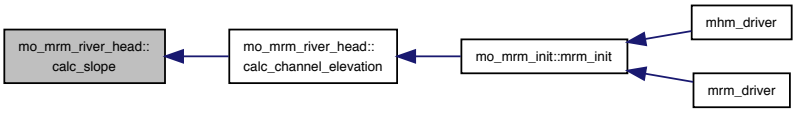
References mo_mrm_net_startup::celllength(), mo_common_variables::iflag_cordinate_sys, and mo_mrm_net_↩ startup::movedownonecell().

Referenced by calc_channel_elevation().

Here is the call graph for this function:



Here is the caller graph for this function:



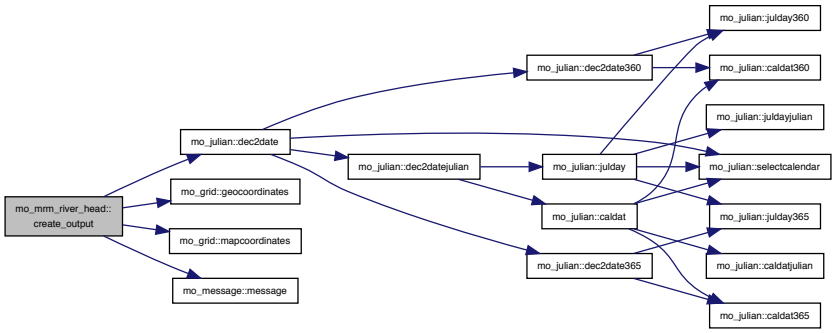
16.59.1.5 create_output()

```
subroutine, public mo_mrm_river_head::create_output (
            integer(i4), intent(in) iBasin,
            character(256), intent(in) OutPath )
```

References mo_common_variables::contact, mo_common_variables::conventions, mo_julian::dec2date(), mo_common_mhm_mrm_variables::evalper, mo_grid::geocoordinates(), mo_common_variables::history, mo_grid::mapcoordinates(), mo_message::message(), mo_common_variables::mhm_details, nc_riverhead, nc_time, mo_common_constants::nodata_dp, mo_common_variables::project_details, mo_common_variables::setup_description, mo_common_variables::simulation_type, sum_counter, time_counter, and mo_file::version.

Referenced by mo_mrm_init::mrm_init().

Here is the call graph for this function:



Here is the caller graph for this function:

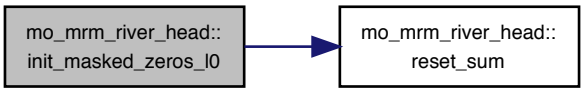
16.59.1.6 init_masked_zeros_l0()

```
subroutine, public mo_mrm_river_head::init_masked_zeros_l0 (
            integer(i4), intent(in) iBasin,
            real(dp), dimension(:), intent(inout), allocatable data )
```

References mo_common_variables::level0, mo_common_constants::nodata_dp, and reset_sum().

Referenced by mo_mrm_init::mrm_init().

Here is the call graph for this function:



Here is the caller graph for this function:

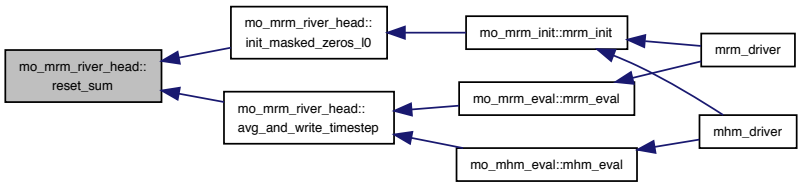
16.59.1.7 reset_sum()

```
subroutine, private mo_mrm_river_head::reset_sum (
             integer(i4), intent(in) iBasin,
             real(dp), dimension(:), intent(inout) data )  [private]
```

References mo_common_variables::level0.

Referenced by avg_and_write_timestep(), and init_masked_zeros_l0().

Here is the caller graph for this function:



16.59.2 Variable Documentation

16.59.2.1 nc_riverhead

```
type(ncvariable), dimension(:), allocatable mo_mrm_river_head::nc_riverhead  [private]
```

Referenced by avg_and_write_timestep(), and create_output().

16.59.2.2 nc_time

```
type(ncvariable), dimension(:), allocatable mo_mrm_river_head::nc_time  [private]
```

Referenced by avg_and_write_timestep(), and create_output().

16.59.2.3 sum_counter

```
integer(i4), dimension(:), allocatable mo_mrm_river_head::sum_counter  [private]
```

Referenced by avg_and_write_timestep(), calc_river_head(), and create_output().

16.59.2.4 time_counter

```
integer(i4), dimension(:), allocatable mo_mrm_river_head::time_counter  [private]
```

Referenced by avg_and_write_timestep(), and create_output().

16.60 mo_mrm_routing Module Reference

Performs runoff routing for mHM at level L11.

Functions/Subroutines

- subroutine, public mrm_routing (read_states, processCase, global_routing_param, L1_total_runoff, L1_↩ areaCell, L1_L11_Id, L11_areaCell, L11_L1_Id, L11_netPerm, L11_fromN, L11_toN, L11_nOutlets, timestep, tsRoutFactor, nNodes, nInflowGauges, InflowGaugeIndexList, InflowGaugeHeadwater, InflowGaugeNode↩ List, InflowDischarge, nGauges, gaugeIndexList, gaugeNodeList, map_flag, L11_length, L11_slope, L11_↩ FracFPimp, L11_C1, L11_C2, L11_qOut, L11_qTIN, L11_qTR, L11_qMod, GaugeDischarge)

  route water given runoff
- subroutine l11_runoff_acc (qAll, efecArea, L1_L11_Id, L11_areaCell, L11_L1_Id, TS, map_flag, qAcc)

  total runoff accumulation at L11.
- subroutine add_inflow (nInflowGauges, InflowIndexList, InflowHeadwater, InflowNodeList, QInflow, qOut)

  Adds inflow discharge to the runoff produced at the cell where the inflow is occurring.
- subroutine l11_routing (nNodes, nLinks, netPerm, netLink_fromN, netLink_toN, netLink_C1, netLink_C2, netNode_qOUT, nInflowGauges, InflowHeadwater, InflowNodeList, netNode_qTIN, netNode_qTR, netNode↩ _Qmod)

  Performs runoff routing for mHM at L11 upscaled network (Routing Network).

16.60.1 Detailed Description

Performs runoff routing for mHM at level L11.

This module performs flood routing at a given time step through the stream network at level L11 to the sink cell. The Muskingum flood routing algorithm is used.

Authors

Luis Samaniego

Date

Dec 2012

16.60.2 Function/Subroutine Documentation

16.60.2.1 add_inflow()

```
subroutine mo_mrm_routing::add_inflow (
             integer(i4), intent(in) nInflowGauges,
             integer(i4), dimension(:), intent(in) InflowIndexList,
             logical, dimension(:), intent(in) InflowHeadwater,
             integer(i4), dimension(:), intent(in) InflowNodeList,
             real(dp), dimension(:), intent(in) QInflow,
             real(dp), dimension(:), intent(inout) qOut )
```

Adds inflow discharge to the runoff produced at the cell where the inflow is occurring.

If a inflow gauge is given, then this routine is adding the values to the runoff produced at the grid cell where the inflow is happening. The values are not directly added to the river network. If this cell is not a headwater then the streamflow produced upstream will be neglected.

Parameters

in	*integer(i4) :: nInflowGauges*	[-] number of inflow points
in	*integer(i4), dimension(:) :: InflowIndexList*	[-] index of inflow points
in	*logical, dimension(:) :: InflowHeadwater*	[-] if to consider headwater cells of inflow gauge
in	*integer(i4), dimension(:) :: InflowNodeList*	[-] L11 ID of inflow points
in	*real(dp), dimension(:) :: QInflow*	[m3 s-1] inflowing water
in,out	*real(dp), dimension(:) :: qOut*	[m3 s-1] Series of attenuated runoff

Authors

Stephan Thober & Matthias Zink

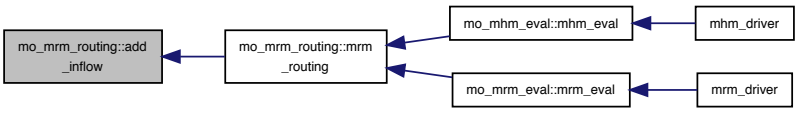
Date

Jul 2016

References mo_kind::dp, and mo_kind::i4.

Referenced by mrm_routing().

Here is the caller graph for this function:



16.60.2.2 l11_routing()

```
subroutine mo_mrm_routing::l11_routing (
            integer(i4), intent(in) nNodes,
            integer(i4), intent(in) nLinks,
            integer(i4), dimension(:), intent(in) netPerm,
            integer(i4), dimension(:), intent(in) netLink_fromN,
            integer(i4), dimension(:), intent(in) netLink_toN,
            real(dp), dimension(:), intent(in) netLink_C1,
            real(dp), dimension(:), intent(in) netLink_C2,
            real(dp), dimension(:), intent(in) netNode_qOUT,
            integer(i4), intent(in) nInflowGauges,
            logical, dimension(:), intent(in) InflowHeadwater,
            integer(i4), dimension(:), intent(in) InflowNodeList,
            real(dp), dimension(:, :), intent(inout) netNode_qTIN,
            real(dp), dimension(:, :), intent(inout) netNode_qTR,
            real(dp), dimension(nnodes), intent(out) netNode_Qmod )
```

Performs runoff routing for mHM at L11 upscaled network (Routing Network).

Hydrograph routing is carried out with the Muskingum algorithm [7]. This simplification of the St. Venant equations is justified in mHM because the potential areas of application of this model would hardly exhibit abruptly changing hydrographs with supercritical flows. The discharge leaving the river reach located on cell i $Q_i^1(t)$ at time step t can be determined by

$$Q_i^1(t) = Q_i^1(t-1) + c_1\left(Q_i^0(t-1) - Q_i^1(t-1)\right) + c_2\left(Q_i^0(t) - Q_i^0(t-1)\right)$$

with

$$Q_i^0(t) = Q_{i'}(t) + Q_{i'}^1(t)$$

$$c_1 = \frac{\Delta t}{\kappa(1-\xi) + \frac{\Delta t}{2}}$$

$$c_2 = \frac{\frac{\Delta t}{2} - \kappa\xi}{\kappa(1-\xi) + \frac{\Delta t}{2}}$$

where Q_i^0 and Q_i^1 denote the discharge entering and leaving the river reach located on cell i respectively. $Q_{i'}$ is the contribution from the upstream cell i'. κ Muskingum travel time parameter. ξ Muskingum attenuation parameter. Δt time interval in hours. t Time index for each Δt interval. To improve performance, a routing sequence "netPerm" is required. This permutation is determined in the mo_init_mrm routine.

TODO: add description

Parameters

in	*integer(i4) :: nNodes*	number of network nodes = nCells1
in	*integer(i4) :: nLinks*	number of stream segment (reaches)
in	*integer(i4), dimension(:) :: netPerm*	routing order of a given basin (permutation)
in	*integer(i4), dimension(:) :: netLink_fromN*	from node
in	*integer(i4), dimension(:) :: netLink_toN*	to node
in	*real(dp), dimension(:) :: netLink_C1*	routing parameter C1 ([7] p. 25-41)
in	*real(dp), dimension(:) :: netLink_C2*	routing parameters C2 (id)
in	*real(dp), dimension(:) :: netNode_qOUT*	Total outflow from cells (given basin) L11 at time tt in [m3 s-1]
in	*integer(i4) :: nInflowGauges*	[-] number of inflow points
in	*logical, dimension(:) :: InflowHeadwater*	[-] if to consider headwater cells of inflow gauge
in	*integer(i4), dimension(:) :: InflowNodeList*	[-] L11 ID of inflow points
in,out	*real(dp), dimension(:, :) :: netNode_qTIN*	[m3 s-1] Total inputs at t-1 and t
in,out	*real(dp), dimension(:, :) :: netNode_qTR*	[m3 s-1] Transformed outflow leaving node I (Muskingum)
out	*real(dp), dimension(nNodes) :: netNode_Qmod*	[m3 s-1] Simulated routed discharge

Authors

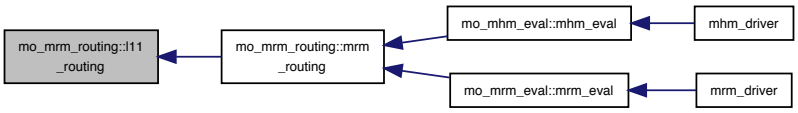
Luis Samaniego

Date

Dec 2005

Referenced by mrm_routing().

Here is the caller graph for this function:



16.60.2.3 l11_runoff_acc()

```
subroutine mo_mrm_routing::l11_runoff_acc (
             real(dp), dimension(:), intent(in) qAll,
             real(dp), dimension(:), intent(in) efecArea,
             integer(i4), dimension(:), intent(in) L1_L11_Id,
             real(dp), dimension(:), intent(in) L11_areaCell,
             integer(i4), dimension(:), intent(in) L11_L1_Id,
             integer(i4), intent(in) TS,
             logical, intent(in) map_flag,
             real(dp), dimension(:), intent(out) qAcc )
```

total runoff accumulation at L11.

Upscales runoff in space from L1 to L11 if routing resolution is higher than hydrology resolution (map_flag equals .true.) or downscales runoff from L1 to L11 if routing resolution is lower than hydrology resolution.

Parameters

in	*real(dp), dimension(:) :: qall*	total runoff L1 [mm tst-1]
in	*real(dp), dimension(:) :: efecarea*	effective area in [km2] at Level 1
in	*integer(i4), dimension(:) :: L1_L11_Id*	L11 Ids mapped on L1
in	*real(dp), dimension(:) :: L11_areacell*	effective area in [km2] at Level 11
in	*integer(i4), dimension(:) :: L11_L1_Id*	L1 Ids mapped on L11
in	*integer(i4) :: TS*	time step in [s]
in	*logical :: map_flag*	Flag indicating whether routing resolution is higher than hydrologic one
out	*real(dp), dimension(:) :: qAcc*	aggregated runoff at L11 [m3 s-1]

Authors

Luis Samaniego

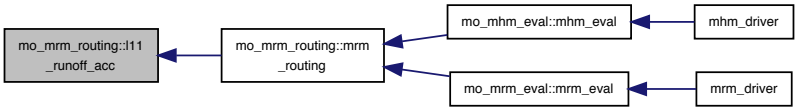
Date

Jan 2013

References mo_common_constants::hoursecs, and mo_common_constants::nodata_dp.

Referenced by mrm_routing().

Here is the caller graph for this function:



16.60.2.4 mrm_routing()

```
subroutine, public mo_mrm_routing::mrm_routing (
            logical, intent(in) read_states,
            integer(i4), intent(in) processCase,
            real(dp), dimension(:), intent(in) global_routing_param,
            real(dp), dimension(:), intent(in) L1_total_runoff,
            real(dp), dimension(:), intent(in) L1_areaCell,
            integer(i4), dimension(:), intent(in) L1_L11_Id,
            real(dp), dimension(:), intent(in) L11_areaCell,
            integer(i4), dimension(:), intent(in) L11_L1_Id,
            integer(i4), dimension(:), intent(in) L11_netPerm,
            integer(i4), dimension(:), intent(in) L11_fromN,
```

```
integer(i4), dimension(:), intent(in) L11_toN,
integer(i4), intent(in) L11_nOutlets,
integer(i4), intent(in) timestep,
real(dp), intent(in) tsRoutFactor,
integer(i4), intent(in) nNodes,
integer(i4), intent(in) nInflowGauges,
integer(i4), dimension(:), intent(in) InflowGaugeIndexList,
logical, dimension(:), intent(in) InflowGaugeHeadwater,
integer(i4), dimension(:), intent(in) InflowGaugeNodeList,
real(dp), dimension(:), intent(in) InflowDischarge,
integer(i4), intent(in) nGauges,
integer(i4), dimension(:), intent(in) gaugeIndexList,
integer(i4), dimension(:), intent(in) gaugeNodeList,
logical, intent(in) map_flag,
real(dp), dimension(:), intent(in) L11_length,
real(dp), dimension(:), intent(in) L11_slope,
real(dp), dimension(:), intent(in) L11_FracFPimp,
real(dp), dimension(:), intent(inout) L11_C1,
real(dp), dimension(:), intent(inout) L11_C2,
real(dp), dimension(:), intent(inout) L11_qOut,
real(dp), dimension(:, :), intent(inout) L11_qTIN,
real(dp), dimension(:, :), intent(inout) L11_qTR,
real(dp), dimension(:), intent(inout) L11_qMod,
real(dp), dimension(:), intent(inout) GaugeDischarge )
```

route water given runoff

This routine first performs mpr for the routing variables if required, then accumulates the runoff to the routing resolution and eventually routes the water in a third step. The last step is repeated multiple times if the routing timestep is smaller than the timestep of the hydrological timestep

Parameters

in	*logical :: read_states*	whether states are derived from restart file
in	*integer(i4) :: processCase*	Process switch for routing
in	*real(dp), dimension(:) :: global_routing_param*	routing parameters
in	*real(dp), dimension(:) :: L1_total_runoff*	total runoff from L1 grid cells
in	*real(dp), dimension(:) :: L1_areaCell*	L1 cell area
in	*integer(i4), dimension(:) :: L1_L11_Id*	L1 cell ids on L11
in	*real(dp), dimension(:) :: L11_areaCell*	L11 cell area
in	*integer(i4), dimension(:) :: L11_L1_Id*	L11 cell ids on L1
in	*integer(i4), dimension(:) :: L11_netPerm*	L11 routing order
in	*integer(i4), dimension(:) :: L11_fromN*	L11 source grid cell order
in	*integer(i4), dimension(:) :: L11_toN*	L11 target grid cell order
in	*integer(i4) :: L11_nOutlets*	L11 number of outlets/sinks
in	*integer(i4) :: timestep*	simulation timestep in [h]
in	*real(dp) :: tsRoutFactor*	factor between routing timestep and hydrological timestep
in	*integer(i4) :: nNodes*	number of nodes
in	*integer(i4) :: nInflowGauges*	number of inflow gauges
in	*integer(i4), dimension(:) :: InflowGaugeIndexList*	index list of inflow gauges
in	*logical, dimension(:) :: InflowGaugeHeadwater*	flag for headwater cell of inflow gauge
in	*integer(i4), dimension(:) :: InflowGaugeNodeList*	gauge node list at L11

Parameters

in	*real(dp), dimension(:) :: InflowDischarge*	inflowing discharge at discharge gauge at current day
in	*integer(i4) :: nGauges*	number of recording gauges
in	*integer(i4), dimension(:) :: gaugeIndexList*	index list for outflow gauges
in	*integer(i4), dimension(:) :: gaugeNodeList*	gauge node list at L11
in	*logical :: map_flag*	flag indicating whether routing resolution iscoarser than hydrologic resolution
in	*real(dp), dimension(:) :: L11_length*	L11 link length
in	*real(dp), dimension(:) :: L11_slope*	L11 slope
in	*real(dp), dimension(:) :: L11_FracFPimp*	L11 fraction of flood plain with impervios cover
in,out	*real(dp), dimension(:) :: L11_C1*	L11 muskingum parameter 1
in,out	*real(dp), dimension(:) :: L11_C2*	L11 muskingum parameter 2
in,out	*real(dp), dimension(:) :: L11_qOut*	total runoff from L11 grid cells
in,out	*real(dp), dimension(:, :) :: L11_qTIN*	L11 inflow to the reach
in,out	*real(dp), dimension(:, :) :: L11_qTR*	L11 routed outflow
in,out	*real(dp), dimension(:) :: L11_qMod*	modelled discharge at each grid cell
in,out	*real(dp), dimension(:) :: GaugeDischarge*	modelled discharge at each gauge

Authors

Stephan Thober

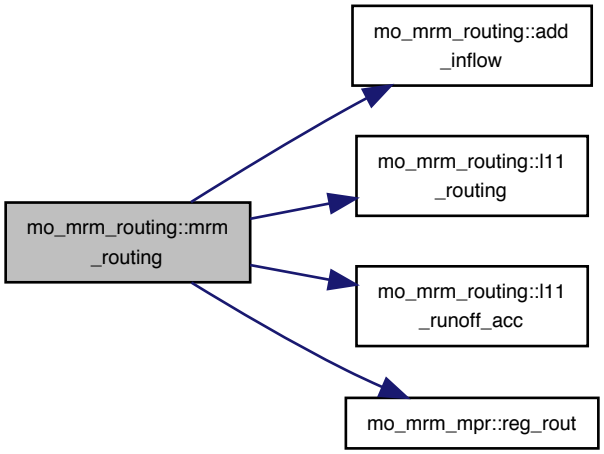
Date

Aug 2015

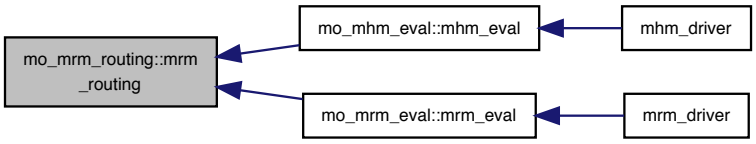
References add_inflow(), mo_mrm_global_variables::is_start, l11_routing(), l11_runoff_acc(), and mo_mrm_mpr↩ ::reg_rout().

Referenced by mo_mhm_eval::mhm_eval(), and mo_mrm_eval::mrm_eval().

Here is the call graph for this function:



Here is the caller graph for this function:



16.61 mo_mrm_signatures Module Reference

Module with calculations for several hydrological signatures.

Functions/Subroutines

- real(dp) function, dimension(size(lags, 1)), public autocorrelation (data, lags, mask)

  Autocorrelation of a given data series.
- real(dp) function, dimension(size(quantiles, 1)), public flowdurationcurve (data, quantiles, mask, concavity_index, mid_segment_slope, mhigh_segment_volume, high_segment_volume, low_segment_volume)

  Flow duration curves.
- subroutine, public limb_densities (data, mask, RLD, DLD)

  Calculates limb densities.
- real(dp) function maximummonthlyflow (data, mask, yr_start, mo_start, dy_start)

  Maximum of average flows per months.
- subroutine, public moments (data, mask, mean_data, stddev_data, median_data, max_data, mean_log, stddev_log, median_log, max_log)

  Moments of data and log-transformed data, e.g. mean and standard deviation.
- real(dp) function, dimension(size(quantiles, 1)), public peakdistribution (data, quantiles, mask, slope_peak_distribution)

  Calculates the peak distribution.
- real(dp) function, public runoffratio (data, basin_area, mask, precip_series, precip_sum, log_data)

  Runoff ratio (accumulated daily discharge [mm/d] / accumulated daily precipitation [mm/d]).
- real(dp) function, public zeroflowratio (data, mask)

  Ratio of zero values to total number of data points.

16.61.1 Detailed Description

Module with calculations for several hydrological signatures.

This module contains calculations for hydrological signatures. It contains:

- Autocorrelation
- Rising and declining limb densities
- Flow duration curves
- Peak distribution

Authors

Remko Nijzink,

Date

March 2014

16.61.2 Function/Subroutine Documentation

16.61.2.1 autocorrelation()

```
real(dp) function, dimension(size(lags, 1)), public mo_mrm_signatures::autocorrelation (
             real(dp), dimension(:), intent(in) data,
             integer(i4), dimension(:), intent(in) lags,
             logical, dimension(size(data, 1)), intent(in), optional mask )
```

Autocorrelation of a given data series.

Calculates the autocorrelation of a data series at given lags. An optional argument for masking data points can be given. The function is basically a wrapper of the function autocorr from the module mo_corr. An optional mask of data points can be specified. ADDITIONAL INFORMATION Used as hydrologic signature with lag 1 in Euser, T., Winsemius, H. C., Hrachowitz, M., Fenicia, F., Uhlenbrook, S., & Savenije, H. H. G. (2013). A framework to assess the realism of model structures using hydrological signatures. Hydrology and Earth System Sciences, 17(5), 1893-1912. doi:10.5194/hess-17-1893-2013

Parameters

in	*real(dp), dimension(:) :: data*	Array of data
in	*integer(i4), dimension(:) :: lags*	Array of lags where autocorrelation is requested
in	*logical, dimension(size(data, 1)), optional :: mask*	Mask for data points givenWorks only with 1d double precision input data.

Authors

Juliane Mai

Date

Jun 2015

16.61.2.2 flowdurationcurve()

```
real(dp) function, dimension(size(quantiles, 1)), public mo_mrm_signatures::flowdurationcurve
(
             real(dp), dimension(:), intent(in) data,
             real(dp), dimension(:), intent(in) quantiles,
             logical, dimension(:), intent(in), optional mask,
             real(dp), intent(out), optional concavity_index,
             real(dp), intent(out), optional mid_segment_slope,
             real(dp), intent(out), optional mhigh_segment_volume,
             real(dp), intent(out), optional high_segment_volume,
             real(dp), intent(out), optional low_segment_volume )
```

Flow duration curves.

Calculates the flow duration curves for a given data vector. The Flow duration curve at a certain quantile x is the data point p where x% of the data points are above the value p. Hence the function percentile of the module mo_percentile is used. But percentile is determining the point p where x% of the data points are below that value. Therfore, the given quantiles are transformed by (1.0-quantile) to get the percentiles of exceedance probabilities. Optionally, the concavity index CI can be calculated [Zhang2014]. CI is defined by

$$CI = \frac{q_{10\%} - q_{99\%}}{q_{1\%} - q_{99\%}}$$

where q_x is the data point where x% of the data points are above that value. Hence, exceedance probabilities are used. Optionally, the FDC mid-segment slope FDC_{MSS} as used by Shafii et. al (2014) can be returned. The FDC_{MSS} is defined as

$$FDC_{MSS} = \log(q_{m_1}) - \log(q_{m_2})$$

where m_1 and m_2 are the lowest and highest flow exceedance probabilities within the midsegment of FDC. The settings $m_1 = 0.2$ and 0.7 are used by Shafii et. al (2014) and are implemented like that. Optionally, the FDC medium high-segment volume FDC_{MHSV} as used by Shafii et. al (2014) can be returned. The FDC_{MHSV} is defined as

$$FDC_{MHSV} = \sum_{h=1}^{H} q_h$$

where $h = 1,2,...,H$ are flow indeces located within the high-flow segment (exceedance probabilities lower than m_1). H is the index of the maximum flow. The settings $m_1 = 0.2$ is used here to be consistent with the definitions of the low-segment (0.7-1.0) and the mid-segment (0.2-0.7). Optionally, the FDC high-segment volume FDC_{HSV} as used by Shafii et. al (2014) can be returned. The FDC_{HSV} is defined as

$$FDC_{HSV} = \sum_{h=1}^{H} q_h$$

where $h = 1,2,...,H$ are flow indeces located within the high-flow segment (exceedance probabilities lower than m_1). H is the index of the maximum flow. The settings $m_1 = 0.02$ is used by Shafii et. al (2014) and is implemented like that. Optionally, the FDC low-segment volume FDC_{LSV} as used by Shafii et. al (2014) can be returned. The FDC_{LSV} is defined as

$$FDC_{LSV} = -\sum_{l=1}^{L} (\log(q_l) - \log(q_L))$$

where $l = 1,2,...,L$ are flow indeces located within the low-flow segment (exceedance probabilities larger than m_1). L is the index of the minimum flow. The settings $m_1 = 0.7$ is used by Shafii et. al (2014) and is implemented like that. An optional mask of data points can be specified. ADDITIONAL INFORMATION Thresholds in mid_segment_slope, mhigh_segment_volume, high_segment_volume, low_segment_volume are hard coded. FDC is used as hydrologic signature (quantiles not specified) in Euser, T., Winsemius, H. C., Hrachowitz, M., Fenicia, F., Uhlenbrook, S., & Savenije, H. H. G. (2013). A framework to assess the realism of model structures using hydrological signatures. Hydrology and Earth System Sciences, 17(5), 1893-1912. doi:10.5194/hess-17-1893-2013 Concavity Index used as hydrologic signature in Zhang, Y., Vaze, J., Chiew, F. H. S., Teng, J., & Li, M. (2014). Predicting hydrological signatures in ungauged catchments using spatial interpolation, index model, and rainfall-runoff modelling. Journal of Hydrology, 517(C), 936-948. doi:10.1016/j.jhydrol.2014.06.032 Concavity index is defined using exceedance probabilities by Sauquet, E., & Catalogne, C. (2011). Comparison of catchment grouping methods for flow duration curve estimation at ungauged sites in France. Hydrology and Earth System Sciences, 15(8), 2421-2435. doi:10.5194/hess-15-2421-2011 mid_segment_slope, high_segment_volume, low_segment_volume used as hydrologic signature in Shafii, M., & Tolson, B. A. (2015). Optimizing hydrological consistency by incorporating hydrological signatures into model calibration objectives. Water Resources Research, 51(5), 3796-3814. doi:10.1002/2014WR016520

Parameters

in	*real(dp), dimension(:) :: data*	data series
in	*real(dp), dimension(:) :: quantiles*	Percentages of exceedance (x-axis of FDC)
in	*logical, dimension(:), optional :: mask*	mask of data array
out	*real(dp), optional :: concavity_index*	concavity index as defined by Sauquet et al. (2011)

Parameters

out	*real(dp), optional :: mid_segment_slope*	mid-segment slope as defined by Shafii et al. (2014)
out	*real(dp), optional :: mhigh_segment_volume*	medium high-segment volume
out	*real(dp), optional :: high_segment_volume*	high-segment volume as defined by Shafii et al. (2014)
out	*real(dp), optional :: low_segment_volume*	low-segment volume as defined by Shafii et al. (2014)

Returns

real(dp), dimension(size(quantiles,1)) :: FlowDurationCurve — Flow Duration Curve value at resp. quantile

Authors

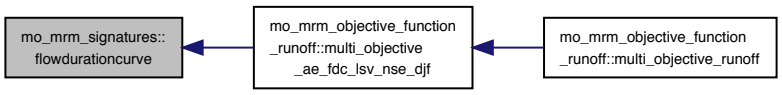
Remko Nijzink, Juliane Mai

Date

March 2014

Referenced by mo_mrm_objective_function_runoff::multi_objective_ae_fdc_lsv_nse_djf().

Here is the caller graph for this function:



16.61.2.3 limb_densities()

```
subroutine, public mo_mrm_signatures::limb_densities (
             real(dp), dimension(:), intent(in) data,
             logical, dimension(size(data, 1)), intent(in), optional mask,
             real(dp), intent(out), optional RLD,
             real(dp), intent(out), optional DLD )
```

Calculates limb densities.

Calculates rising and declinging limb densities. The peaks of the given series are first determined by looking for points where preceding and subsequent datapoint are lower. Second, the number of datapoints with rising values (nrise) and declining values (ndecline) are counted basically by comparing neighbors. The duration the data increase (nrise) divided by the number of peaks (npeaks) gives the rising limb density RLD

$$RLD = t_{rise}/n_{peak}$$

whereas the duration the data decrease (ndecline) divided by the number of peaks (npeaks) gives the declining limb density DLD

$$DLD = t_{fall}/n_{peak}.$$

An optional mask of data points can be specified. ADDITIONAL INFORMATION Rising limb density used as hydrologic signature in Euser, T., Winsemius, H. C., Hrachowitz, M., Fenicia, F., Uhlenbrook, S., & Savenije, H. H. G. (2013). A framework to assess the realism of model structures using hydrological signatures. Hydrology and Earth System Sciences, 17(5), 1893-1912. doi:10.5194/hess-17-1893-2013

Parameters

in	*real(dp), dimension(:) :: data*	data series
in	*logical, dimension(size(data, 1)), optional :: mask*	mask for data series
out	*real(dp), optional :: RLD*	rising limb density
out	*real(dp), optional :: DLD*	declining limb density

Authors

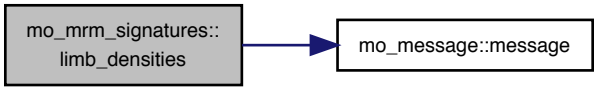
Remko Nijzink

Date

March 2014

References mo_message::message().

Here is the call graph for this function:



16.61.2.4 maximummonthlyflow()

```
real(dp) function mo_mrm_signatures::maximummonthlyflow (
             real(dp), dimension(:), intent(in) data,
             logical, dimension(size(data, 1)), intent(in), optional mask,
             integer(i4), intent(in), optional yr_start,
             integer(i4), intent(in), optional mo_start,
             integer(i4), intent(in), optional dy_start )
```

Maximum of average flows per months.

Maximum of average flow per month is defined as

$$max_{monthlyflow} = Max(F(i), i = 1, ..12)$$

where $f F(i) $ is the average flow of month i. ADDITIONAL INFORMATION used as hydrologic signature in Shafii, M., & Tolson, B. A. (2015). Optimizing hydrological consistency by incorporating hydrological signatures into model calibration objectives. Water Resources Research, 51(5), 3796-3814. doi:10.1002/2014WR016520

Parameters

in	*real(dp), dimension(:) :: data*	array of data
in	*logical, dimension(size(data, 1)), optional :: mask*	mask for data points given
in	*integer(i4), optional :: yr_start*	year of date of first data point given
in	*integer(i4), optional :: mo_start*	month of date of first data point given (default: 1)
in	*integer(i4), optional :: dy_start*	month of date of first data point given (default: 1)

Returns

real(dp) :: MaximumMonthlyFlow — Maximum of average flow per month Works only with 1d double precision input data. Assumes data are daily values.

Authors

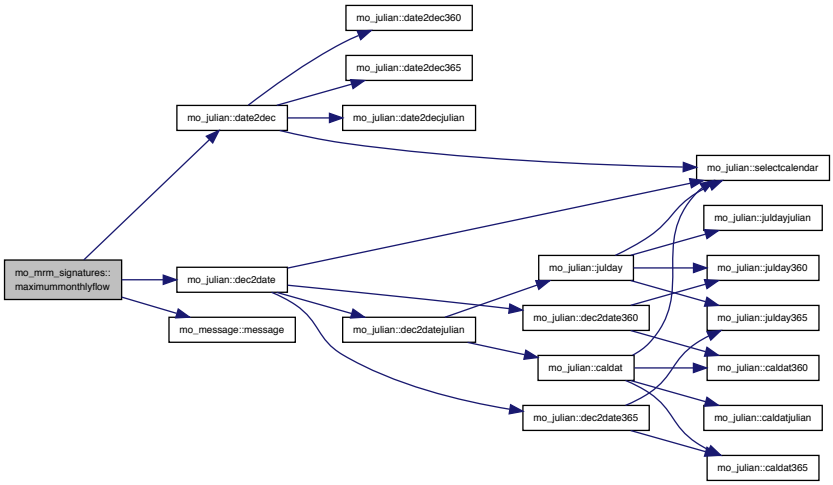
Juliane Mai

Date

Jun 2015

References mo_julian::date2dec(), mo_julian::dec2date(), and mo_message::message().

Here is the call graph for this function:



16.61.2.5 moments()

```
subroutine, public mo_mrm_signatures::moments (
             real(dp), dimension(:), intent(in) data,
             logical, dimension(size(data, 1)), intent(in), optional mask,
             real(dp), intent(out), optional mean_data,
             real(dp), intent(out), optional stddev_data,
             real(dp), intent(out), optional median_data,
             real(dp), intent(out), optional max_data,
             real(dp), intent(out), optional mean_log,
             real(dp), intent(out), optional stddev_log,
             real(dp), intent(out), optional median_log,
             real(dp), intent(out), optional max_log )
```

Moments of data and log-transformed data, e.g. mean and standard deviation.

Returns several moments of data series given, i.e.

- mean of data

- standard deviation of data
- median of data
- maximum/ peak of data
- mean of log-transformed data
- standard deviation of log-transformed data
- median of log-transformed data
- maximum/ peak of log-transformed data An optional mask of data points can be specified. ADDITIONAL INFORMATION mean_log and stddev_log used as hydrologic signature in Zhang, Y., Vaze, J., Chiew, F. H. S., Teng, J., & Li, M. (2014). Predicting hydrological signatures in ungauged catchments using spatial interpolation, index model, and rainfall-runoff modelling. Journal of Hydrology, 517(C), 936-948. doi:10.↩ 1016/j.jhydrol.2014.06.032 mean_data, stddev_data, median_data, max_data, mean_log, and stddev_log used as hydrologic signature in Shafii, M., & Tolson, B. A. (2015). Optimizing hydrological consistency by incorporating hydrological signatures into model calibration objectives. Water Resources Research, 51(5), 3796-3814. doi:10.1002/2014WR016520

Parameters

in	*real(dp), dimension(:) :: data*	array of data
in	*logical, dimension(size(data, 1)), optional :: mask*	mask for data points given
out	*real(dp), optional :: mean_data*	mean of data
out	*real(dp), optional :: stddev_data*	standard deviation of data
out	*real(dp), optional :: median_data*	median of data
out	*real(dp), optional :: max_data*	maximum/ peak of data
out	*real(dp), optional :: mean_log*	mean of log-transformed data
out	*real(dp), optional :: stddev_log*	standard deviation of log-transformed data
out	*real(dp), optional :: median_log*	median of log-transformed data
out	*real(dp), optional :: max_log*	maximum/ peak of log-transformed dataWorks only with 1d double precision input data.

Authors

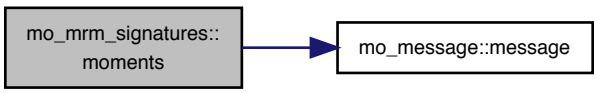
Juliane Mai

Date

Jun 2015

References mo_message::message().

Here is the call graph for this function:



16.61.2.6 peakdistribution()

```
real(dp) function, dimension(size(quantiles, 1)), public mo_mrm_signatures::peakdistribution (
            real(dp), dimension(:), intent(in) data,
            real(dp), dimension(:), intent(in) quantiles,
            logical, dimension(size(data, 1)), intent(in), optional mask,
            real(dp), intent(out), optional slope_peak_distribution )
```

Calculates the peak distribution.

First, the peaks of the time series given are identified. For the peak distribution only this subset of data points are considered. Second, the peak distribution at the quantiles given is calculated. Calculates the peak distribution at the quantiles given using mo_percentile. Since the exceedance probabilities are usually used in hydrology the function percentile is used with (1.0-quantiles). Optionally, the slope of the peak distribution between 10th and 50th percentile, i.e.

$$slope = \frac{\text{peak\_data}_{0.1} - \text{peak\_data}_{0.5}}{0.9 - 0.5}$$

can be returned. An optional mask for the data points can be given. ADDITIONAL INFORMATION slope_peak_↩ distribution used as hydrologic signature in Euser, T., Winsemius, H. C., Hrachowitz, M., Fenicia, F., Uhlenbrook, S., & Savenije, H. H. G. (2013). A framework to assess the realism of model structures using hydrological signatures. Hydrology and Earth System Sciences, 17(5), 1893-1912. doi:10.5194/hess-17-1893-2013

Parameters

in	*real(dp), dimension(:) :: data*	data array
in	*real(dp), dimension(:) :: quantiles*	requested quantiles for distribution
in	*logical, dimension(size(data, 1)), optional :: mask*	mask of data array
out	*real(dp), optional :: slope_peak_distribution*	slope of the Peak distribution between10th and 50th percentile

Returns

real(dp), dimension(size(quantiles,1)) :: PeakDistribution — Distribution of peak values at resp. quantiles

Authors

Remko Nijzink

Date

March 2014

16.61.2.7 runoffratio()

```
real(dp) function, public mo_mrm_signatures::runoffratio (
            real(dp), dimension(:), intent(in) data,
            real(dp), intent(in) basin_area,
            logical, dimension(size(data, 1)), intent(in), optional mask,
            real(dp), dimension(size(data, 1)), intent(in), optional precip_series,
            real(dp), intent(in), optional precip_sum,
            logical, intent(in), optional log_data )
```

Runoff ratio (accumulated daily discharge [mm/d] / accumulated daily precipitation [mm/d]).

The runoff ratio is defined as

$$runoff_ratio = \frac{\sum_{t=1}^{N} q_t}{\sum_{t=1}^{N} p_t}$$

where p_t and q_t are precipitation and discharge, respectively. Therefore, precipitation over the entire basin is required and both discharge and precipitation have to be converted to the same units [mm/d]. Input discharge is given in [m∗∗3/s] as this is mHM default while precipitation has to be given in [mm/km∗∗2 / day]. Either "precip_sum" or "precip_series" has to be specified. If "precip_series" is used the optional mask is also applied to precipitation values. The "precip_sum" is the accumulated "precip_series". Optionally, a mask for the data (=discharge) can be given. If optional "log_data" is set to .true. the runoff ratio will be calculated as

$$runoff\_ratio = \frac{\sum_{t=1}^{N} \log(q_t)}{\sum_{t=1}^{N} p_t}$$

where p_t and q_t are precipitation and discharge, respectively. ADDITIONAL INFORMATION

Returns

real(dp), dimension(size(lags,1)) :: RunoffRation — Ratio of discharge and precipitation Used as hydrologic signature in Shafii, M., & Tolson, B. A. (2015). Optimizing hydrological consistency by incorporating hydrological signatures into model calibration objectives. Water Resources Research, 51(5), 3796-3814. doi:10.↩ 1002/2014WR016520

Parameters

in	*real(dp), dimension(:) :: data*	array of data [m∗∗3/s]
in	*real(dp) :: basin_area*	area of basin [km∗∗2]
in	*logical, dimension(size(data, 1)), optional :: mask*	mask for data points given
in	*real(dp), dimension(size(data, 1)), optional :: precip_series*	daily precipitation values [mm/km∗∗2 / day]
in	*real(dp), optional :: precip_sum*	sum of daily precip. values of whole period[mm/km∗∗2 / day]
in	*logical, optional :: log_data*	ratio using logarithmic dataWorks only with 1d double precision input data.

Authors

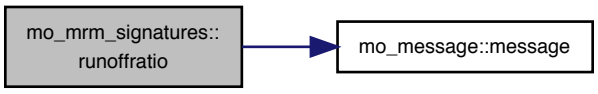
Juliane Mai

Date

Jun 2015

References mo_message::message().

Here is the call graph for this function:



16.61.2.8 zeroflowratio()

```
real(dp) function, public mo_mrm_signatures::zeroflowratio (
            real(dp), dimension(:), intent(in) data,
            logical, dimension(size(data, 1)), intent(in), optional mask )
```

Ratio of zero values to total number of data points.

An optional mask of data points can be specified. ADDITIONAL INFORMATION

Returns

real(dp), dimension(size(lags,1)) :: ZeroFlowRatio — Ratio of zero values to total number of data points Used as hydrologic signature in Zhang, Y., Vaze, J., Chiew, F. H. S., Teng, J., & Li, M. (2014). Predicting hydrological signatures in ungauged catchments using spatial interpolation, index model, and rainfall-runoff modelling. Journal of Hydrology, 517(C), 936-948. doi:10.1016/j.jhydrol.2014.06.032

Parameters

in	*real(dp), dimension(:) :: data*	array of data
in	*logical, dimension(size(data, 1)), optional :: mask*	mask for data points givenWorks only with 1d double precision input data.

Authors

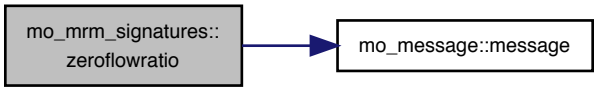
Juliane Mai

Date

Jun 2015

References mo_message::message().

Here is the call graph for this function:



16.62 mo_mrm_write Module Reference

write of discharge and restart files

Functions/Subroutines

- subroutine, public mrm_write

  write discharge and restart files
- subroutine write_configfile

This modules writes the results of the configuration into an ASCII-file.

- subroutine write_daily_obs_sim_discharge (Qobs, Qsim)

  Write a file for the daily observed and simulated discharge timeseries during the evaluation period for each gauging station.

- subroutine, public mrm_write_output_fluxes (iBasin, nCells, timeStep_model_outputs, warmingDays, newTime, nTimeSteps, nTStepDay, tt, day, month, year, timestep, mask11, L11_qmod)

  write fluxes to netcdf output files

- subroutine, public mrm_write_optifile (best_OF, best_paramSet, param_names)

  Write briefly final optimization results.

- subroutine, public mrm_write_optinamelist (parameters, maskpara, parameters_name)

  Write final, optimized parameter set in a namelist format.

Variables

- integer(i4) day_counter
- integer(i4) month_counter
- integer(i4) year_counter
- integer(i4) average_counter
- type(outputdataset) nc

16.62.1 Detailed Description

write of discharge and restart files

This module contains the subroutines for writing the discharge files and optionally the restart files.

Authors

Stephan Thober

Date

Aug 2015

16.62.2 Function/Subroutine Documentation

16.62.2.1 mrm_write()

```
subroutine, public mo_mrm_write::mrm_write ( )
```

write discharge and restart files

First, this subroutine calls the writing or restart files that only succeeds if it happens after the write of mHM restart files because mHM restart files must exist. Second, simulated discharge is aggregated to the daily scale and then written to file jointly with observed discharge

Authors

Juliane Mai, Rohini Kumar & Stephan Thober

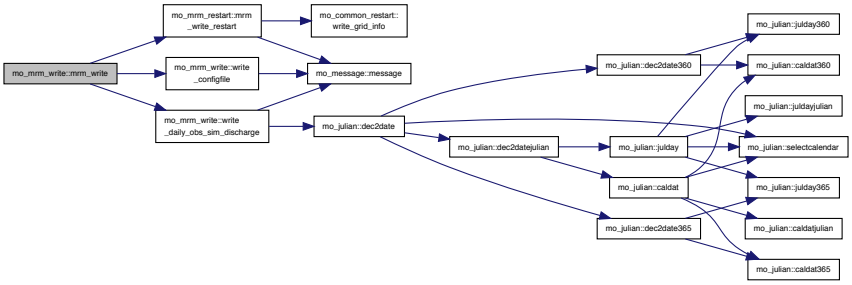
Date

Aug 2015

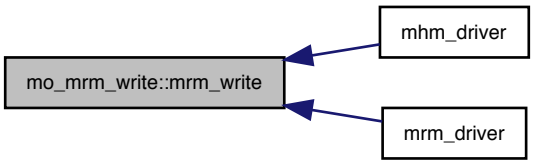
References mo_mrm_global_variables::basin_mrm, mo_common_variables::dirrestartout, mo_common_mhm_↩ mrm_variables::evalper, mo_mrm_global_variables::gauge, mo_common_mhm_mrm_variables::mrm_coupling_↩ mode, mo_mrm_global_variables::mrm_runoff, mo_mrm_restart::mrm_write_restart(), mo_common_variables↩ ::nbasins, mo_mrm_global_variables::ngaugestotal, mo_common_mhm_mrm_variables::ntstepday, mo_common↩ _mhm_mrm_variables::simper, mo_common_mhm_mrm_variables::warmingdays, write_configfile(), write_daily_↩ obs_sim_discharge(), and mo_common_variables::write_restart.

Referenced by mhm_driver(), and mrm_driver().

Here is the call graph for this function:



Here is the caller graph for this function:



16.62.2.2 mrm_write_optifile()

```
subroutine, public mo_mrm_write::mrm_write_optifile (
             real(dp), intent(in) best_OF,
             real(dp), dimension(:), intent(in) best_paramSet,
             character(len = *), dimension(:), intent(in) param_names )
```

Write briefly final optimization results.

Write overall best objective function and the best optimized parameter set to a file_opti.

Parameters

in	*real(dp) :: best_OF*	best objective function value as returnedby the optimization routine
in	*real(dp), dimension(:) :: best_paramSet*	best associated global parameter setCalled only when optimize is .TRUE.
in	*character(len = ∗), dimension(:) :: param_names*	

Authors

David Schaefer

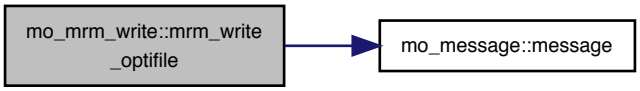
Date

July 2013

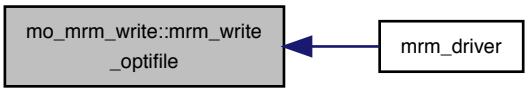
References mo_common_variables::dirconfigout, mo_common_mhm_mrm_file::file_opti, mo_message↩ ::message(), and mo_common_mhm_mrm_file::uopti.

Referenced by mrm_driver().

Here is the call graph for this function:



Here is the caller graph for this function:



16.62.2.3 mrm_write_optinamelist()

```
subroutine, public mo_mrm_write::mrm_write_optinamelist (
             real(dp), dimension(:, :), intent(in) parameters,
             logical, dimension(size(parameters, 1)), intent(in) maskpara,
             character(len = ∗), dimension(size(parameters, 1)), intent(in) parameters_name )
```

Write final, optimized parameter set in a namelist format.

Write final, optimized parameter set in a namelist format.

Parameters

in	*real(dp), dimension(:, :) :: parameters*	(min, max, opti)
in	*logical, dimension(size(parameters, 1)) :: maskpara*	.true. if parameter was calibrated
in	*character(len = ∗), dimension(size(parameters, 1)) :: parameters_name*	clear names of parameters

Authors

Juliane Mai

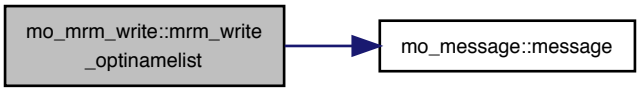
Date

Dec 2013

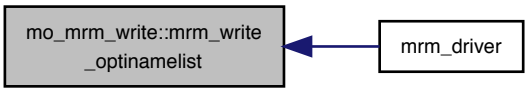
References mo_common_variables::dirconfigout, mo_common_mhm_mrm_file::file_opti_nml, mo_message↩ ::message(), mo_common_variables::processmatrix, and mo_common_mhm_mrm_file::uopti_nml.

Referenced by mrm_driver().

Here is the call graph for this function:



Here is the caller graph for this function:



16.62.2.4 mrm_write_output_fluxes()

```
subroutine, public mo_mrm_write::mrm_write_output_fluxes (
             integer(i4), intent(in) iBasin,
             integer(i4), intent(in) nCells,
             integer(i4), intent(in) timeStep_model_outputs,
             integer(i4), intent(in) warmingDays,
             real(dp), intent(in) newTime,
             integer(i4), intent(in) nTimeSteps,
```

```
integer(i4), intent(in) nTStepDay,
integer(i4), intent(in) tt,
integer(i4), intent(in) day,
integer(i4), intent(in) month,
integer(i4), intent(in) year,
integer(i4), intent(in) timestep,
logical, dimension(:, :), intent(in), pointer mask11,
real(dp), dimension(:), intent(in) L11_qmod )
```

write fluxes to netcdf output files

This subroutine creates a netcdf data set for writing L11_QTIN for different time averages.

Parameters

in	*integer(i4) :: iBasin*	
in	*integer(i4) :: nCells*	
in	*integer(i4) :: timeStep_model_outputs*	timestep of model outputs
in	*integer(i4) :: warmingDays*	number of warming days
in	*real(dp) :: newTime*	julian date of next time step
in	*integer(i4) :: nTimeSteps*	number of total timesteps
in	*integer(i4) :: nTStepDay*	number of timesteps per day
in	*integer(i4) :: tt*	current model timestep
in	*integer(i4) :: day*	current day of the year
in	*integer(i4) :: month*	current month of the year
in	*integer(i4) :: year*	current year
in	*integer(i4) :: timestep*	current model time resolution
in	*logical, dimension(:, :) :: mask11*	mask at level 11
in	*real(dp), dimension(:) :: L11_qMod*	current routed streamflow

Authors

Stephan Thober

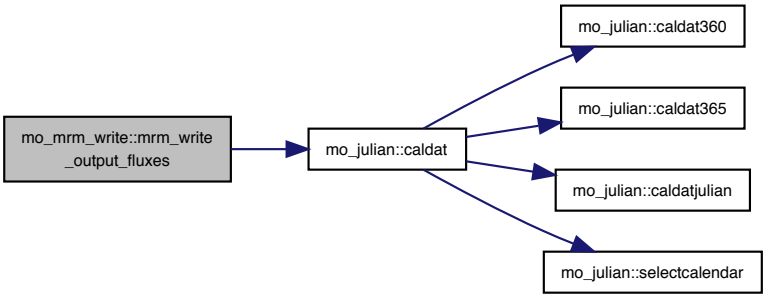
Date

Aug 2015

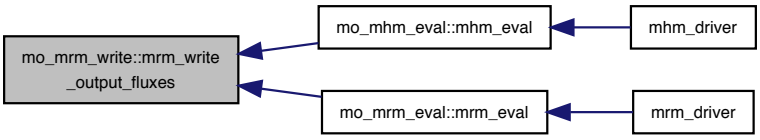
References average_counter, mo_julian::caldat(), day_counter, mo_kind::dp, mo_kind::i4, month_counter, nc, and year_counter.

Referenced by mo_mhm_eval::mhm_eval(), and mo_mrm_eval::mrm_eval().

Here is the call graph for this function:



Here is the caller graph for this function:



16.62.2.5 write_configfile()

```
subroutine mo_mrm_write::write_configfile ( )
```

This modules writes the results of the configuration into an ASCII-file.

TODO: add description

Authors

Christoph Schneider

Date

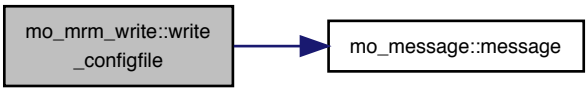
May 2013

References mo_mrm_global_variables::basin_mrm, mo_common_variables::dirconfigout, mo_mrm_global↩ _variables::dirgauges, mo_common_variables::dirlcover, mo_common_variables::dirmorpho, mo_common↩ _variables::dirout, mo_common_variables::dirrestartout, mo_mrm_global_variables::dirtotalrunoff, mo_kind↩ ::dp, mo_common_mhm_mrm_variables::evalper, mo_common_file::file_config, mo_mrm_global_variables↩ ::gauge, mo_common_variables::global_parameters, mo_common_variables::global_parameters_name, mo_↩ kind::i4, mo_mrm_global_variables::inflowgauge, mo_common_variables::l0_basin, mo_mrm_global_variables↩ ::l11_fromn, mo_mrm_global_variables::l11_l1_id, mo_mrm_global_variables::l11_label, mo_mrm_global_↩ variables::l11_length, mo_mrm_global_variables::l11_netperm, mo_mrm_global_variables::l11_rorder, mo_mrm↩

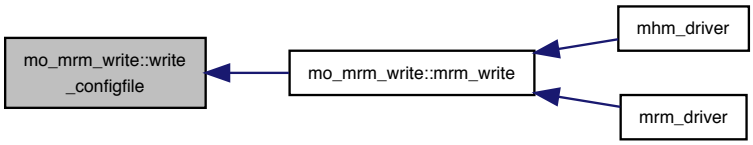
_global_variables::l11_slope, mo_mrm_global_variables::l11_ton, mo_mrm_global_variables::l1_l11_id, mo_common_variables::lc_year_end, mo_common_variables::lc_year_start, mo_common_variables::lcfilename, mo_common_mhm_mrm_variables::lcyearid, mo_common_variables::level0, mo_common_variables::level1, mo_mrm_global_variables::level11, mo_message::message(), mo_common_mhm_mrm_variables::mrm_coupling_mode, mo_common_variables::nbasins, mo_mrm_global_variables::ngaugestotal, mo_mrm_global_variables::ninflowgaugestotal, mo_common_variables::nlcoverscene, mo_common_constants::nodata_dp, mo_common_variables::processmatrix, mo_common_mhm_mrm_variables::read_restart, mo_common_variables::resolutionhydrology, mo_common_mhm_mrm_variables::resolutionrouting, mo_common_mhm_mrm_variables::simper, mo_common_mhm_mrm_variables::timestep, mo_common_file::uconfig, mo_mrm_file::version, mo_common_mhm_mrm_variables::warmper, and mo_common_variables::write_restart.

Referenced by mrm_write().

Here is the call graph for this function:



Here is the caller graph for this function:



16.62.2.6 write_daily_obs_sim_discharge()

```
subroutine mo_mrm_write::write_daily_obs_sim_discharge (
            real(dp), dimension(:, :), intent(in) Qobs,
            real(dp), dimension(:, :), intent(in) Qsim )
```

Write a file for the daily observed and simulated discharge timeseries during the evaluation period for each gauging station.

Write a file for the daily observed and simulated discharge timeseries during the evaluation period for each gauging station

Parameters

in	*real(dp), dimension(:, :) :: Qobs*	daily time series of observed dischargedims = (nModeling_days , nGauges_total)

Parameters

in	*real(dp), dimension(:, :) :: Qsim*	daily time series of modeled dischargedims = (nModeling_days , nGauges_total)

Authors

Rohini Kumar

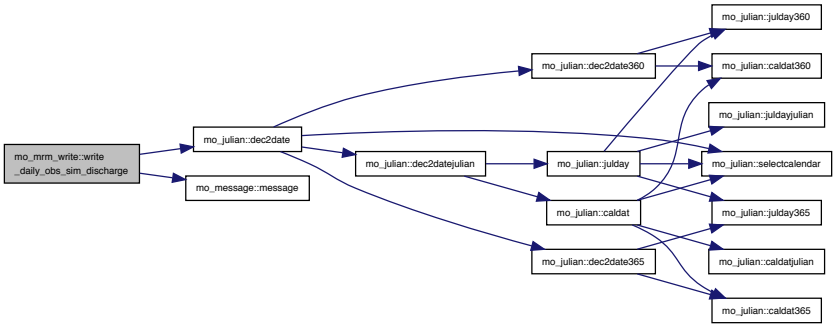
Date

August 2013

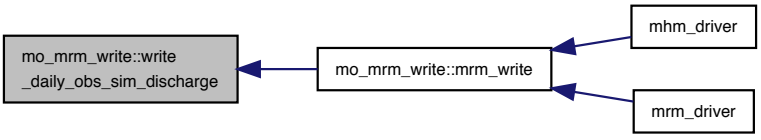
References mo_mrm_global_variables::basin_mrm, mo_julian::dec2date(), mo_common_variables::dirout, mo_common_mhm_mrm_variables::evalper, mo_mrm_file::file_daily_discharge, mo_mrm_global_variables::gauge, mo_message::message(), mo_common_variables::nbasins, mo_mrm_file::ncfile_discharge, and mo_mrm_file::udaily_discharge.

Referenced by mrm_write().

Here is the call graph for this function:



Here is the caller graph for this function:



16.62.3 Variable Documentation

16.62.3.1 average_counter

```
integer(i4) mo_mrm_write::average_counter  [private]
```

Referenced by mrm_write_output_fluxes().

16.62.3.2 day_counter

```
integer(i4) mo_mrm_write::day_counter  [private]
```

Referenced by mrm_write_output_fluxes().

16.62.3.3 month_counter

```
integer(i4) mo_mrm_write::month_counter  [private]
```

Referenced by mrm_write_output_fluxes().

16.62.3.4 nc

```
type(outputdataset) mo_mrm_write::nc  [private]
```

Referenced by mrm_write_output_fluxes().

16.62.3.5 year_counter

```
integer(i4) mo_mrm_write::year_counter  [private]
```

Referenced by mrm_write_output_fluxes().

16.63 mo_mrm_write_fluxes_states Module Reference

Creates NetCDF output for different fluxes and state variables of mHM.

Data Types

- interface outputdataset
- interface outputvariable

Functions/Subroutines

- type(outputvariable) function newoutputvariable (nc, name, dtype, dims, ncells, mask, avg)

  Initialize OutputVariable.
- subroutine updatevariable (self, data)

  Update OutputVariable.
- subroutine writevariabletimestep (self, timestep)

  Write timestep to file.

- type(outputdataset) function newoutputdataset (ibasin, mask, nCells)

  Initialize OutputDataset.
- subroutine updatedataset (self, sidx, eidx, L11_Qmod)

  Update all variables.
- subroutine writetimestep (self, timestep)

  Write all accumulated data.
- subroutine close (self)

  Close the file.
- type(ncdataset) function createoutputfile (ibasin)

  Create and initialize output file.
- subroutine writevariableattributes (var, long_name, unit)

  Write output variable attributes.

16.63.1 Detailed Description

Creates NetCDF output for different fluxes and state variables of mHM.

NetCDF is first initialized and later on variables are put to the NetCDF.

Authors

Matthias Zink

Date

Apr 2013

16.63.2 Function/Subroutine Documentation

16.63.2.1 close()

```
subroutine mo_mrm_write_fluxes_states::close (
            class(outputdataset) self )  [private]
```

Close the file.

Close the file associated with variable of type(OutputDataset)

Authors

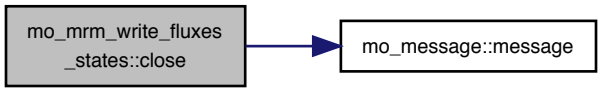
Rohini Kumar & Stephan Thober

Date

August 2013

References mo_common_variables::dirout, and mo_message::message().

Here is the call graph for this function:



16.63.2.2 createoutputfile()

```
type(ncdataset) function mo_mrm_write_fluxes_states::createoutputfile (
             integer(i4), intent(in) ibasin )
```

Create and initialize output file.

Create output file, write all non-dynamic variables and global attributes for the given basin.

Returns

type(NcDataset)

Parameters

in	*integer(i4) :: ibasin*	-> basin id

Authors

David Schaefer

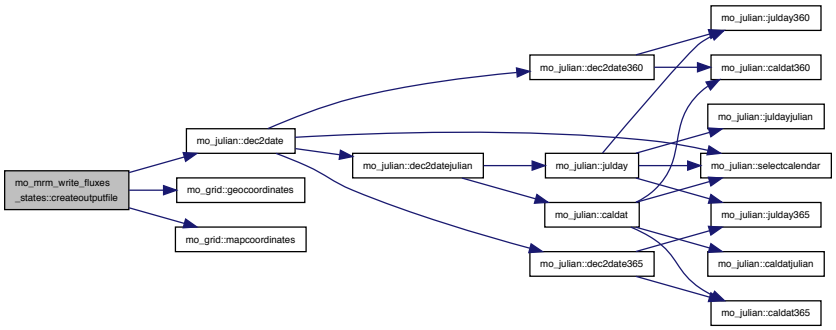
Date

June 2015

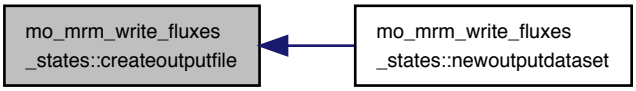
References mo_julian::dec2date(), mo_common_variables::dirout, mo_common_mhm_mrm_variables::evalper, mo_mrm_file::file_mrm_output, mo_grid::geocoordinates(), mo_mrm_global_variables::level11, mo_grid↩ ::mapcoordinates(), mo_common_constants::nodata_dp, and mo_mrm_file::version.

Referenced by newoutputdataset().

Here is the call graph for this function:



Here is the caller graph for this function:



16.63.2.3 newoutputdataset()

```
type(outputdataset) function mo_mrm_write_fluxes_states::newoutputdataset (
            integer(i4), intent(in) ibasin,
            logical, dimension(:, :), intent(in), pointer mask,
            integer(i4), intent(in) nCells )  [private]
```

Initialize OutputDataset.

Create and initialize the output file. If new a new output variable needs to be written, this is the first of two procedures to change (second: updateDataset)

Returns

type(OutputDataset)

Parameters

in	*integer(i4) :: ibasin*	-> basin id
in	*logical, dimension(:, :) :: mask*	
in	*integer(i4) :: nCells*	

Authors

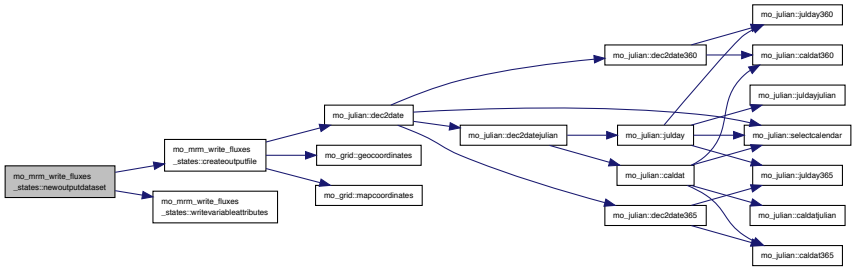
Matthias Zink

Date

Apr 2013

References createoutputfile(), mo_mrm_global_variables::outputflxstate_mrm, and writevariableattributes().

Here is the call graph for this function:



16.63.2.4 newoutputvariable()

```
type(outputvariable) function mo_mrm_write_fluxes_states::newoutputvariable (
            type(ncdataset), intent(in) nc,
            character(*), intent(in) name,
            character(*), intent(in) dtype,
            character(16), dimension(3), intent(in) dims,
            integer(i4), intent(in) ncells,
            logical, dimension(:, :), intent(in), target mask,
            logical, intent(in), optional avg )  [private]
```

Initialize OutputVariable.

TODO: add description

Returns

type(OutputVariable)

Parameters

in	*type(NcDataset) :: nc*	-> NcDataset which contains the variable
in	*character(*) :: name*	
in	*character(*) :: dtype*	

Parameters

in	*character(16), dimension(3) :: dims*	
in	*integer(i4) :: ncells*	-> number of cells in basin
in	*logical, dimension(:, :) :: mask*	
in	*logical, optional :: avg*	-> average the data before writing

Authors

David Schaefer

Date

June 2015

16.63.2.5 updatedataset()

```
subroutine mo_mrm_write_fluxes_states::updatedataset (
             class(outputdataset), intent(inout), target self,
             integer(i4), intent(in) sidx,
             integer(i4), intent(in) eidx,
             real(dp), dimension(:), intent(in) L11_Qmod )
```

Update all variables.

Call the type bound procedure updateVariable for all output variables. If a new output variable needs to be written, this is the second of two procedures to change (first: newOutputDataset)

Parameters

in,out	*class(OutputDataset) :: self*	
in	*integer(i4) :: sidx, eidx*	- start index of the basin related data in L1_* arguments
in	*integer(i4) :: sidx, eidx*	- end index of the basin related data in L1_* arguments
in	*real(dp), dimension(:) :: L11_Qmod*	

Authors

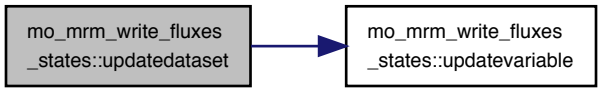
Matthias Zink

Date

Apr 2013

References mo_mrm_global_variables::outputflxstate_mrm, and updatevariable().

Here is the call graph for this function:



16.63.2.6 updatevariable()

```
subroutine mo_mrm_write_fluxes_states::updatevariable (
             class(outputvariable), intent(inout) self,
             real(dp), dimension(:), intent(in) data )  [private]
```

Update OutputVariable.

Add the array given as actual argument to the derived type's component 'data'

Returns

type(OutputVariable)

Parameters

in,out	*class(OutputVariable) :: self*	
in	*real(dp), dimension(:) :: data*	

Authors

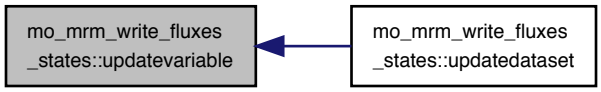
David Schaefer

Date

June 2015

Referenced by updatedataset().

Here is the caller graph for this function:



16.63.2.7 writetimestep()

```
subroutine mo_mrm_write_fluxes_states::writetimestep (
             class(outputdataset), intent(inout), target self,
             integer(i4), intent(in) timestep )
```

Write all accumulated data.

Write all accumulated and potentially averaged data to disk.

Returns

type(OutputVariable)

Parameters

in,out	*class(OutputDataset) :: self*	
in	*integer(i4) :: timestep*	The model timestep to write

Authors

David Schaefer

Date

June 2015

16.63.2.8 writevariableattributes()

```
subroutine mo_mrm_write_fluxes_states::writevariableattributes (
             type(outputvariable), intent(in) var,
             character(*), intent(in) long_name,
             character(*), intent(in) unit )
```

Write output variable attributes.

TODO: add description

Parameters

in	*type(OutputVariable) :: var*	
in	*character(∗) :: long_name, unit*	-> variable name
in	*character(∗) :: long_name, unit*	-> physical unit

Authors

David Schaefer

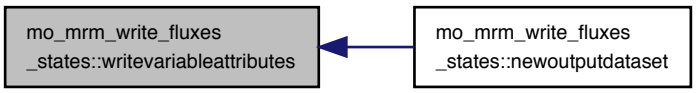
Date

June 2015

References mo_common_constants::nodata_dp.

Referenced by newoutputdataset().

Here is the caller graph for this function:



16.63.2.9 writevariabletimestep()

```
subroutine mo_mrm_write_fluxes_states::writevariabletimestep (
            class(outputvariable), intent(inout) self,
            integer(i4), intent(in) timestep )  [private]
```

Write timestep to file.

Write the content of the derived types's component 'data' to file, average if necessary

Parameters

in,out	*class(OutputVariable) :: self*	
in	*integer(i4) :: timestep*	-> index along the time dimension of the netcdf variable

Authors

David Schafer

Date

June 2015

References mo_common_constants::nodata_dp.

16.64 mo_multi_param_reg Module Reference

Multiscale parameter regionalization (MPR).

Functions/Subroutines

- subroutine, public mpr (mask0, geoUnit0, soilId0, Asp0, gridded_LAI0, LCover0, slope_emp0, y0, Id0, upper_bound1, lower_bound1, left_bound1, right_bound1, n_subcells1, fSealed1, alpha1, degDayInc1, degDayMax1, degDayNoPre1, fAsp1, HarSamCoeff1, PrieTayAlpha1, aeroResist1, surfResist1, fRoots1, kFastFlow1, kSlowFlow1, kBaseFlow1, kPerco1, karstLoss1, soilMoistFC1, soilMoistSat1, soilMoistExp1, jarvis_thresh_c1, tempThresh1, unsatThresh1, sealedThresh1, wiltingPoint1, maxInter1, petLAIcorFactor, parameterset)

  Regionalizing and Upscaling process parameters.
- subroutine baseflow_param (param, geoUnit0, k2_0)

  baseflow recession parameter
- subroutine snow_acc_melt_param (param, fForest1, fIperm1, fPerm1, tempThresh1, degDayNoPre1, deg↩ DayInc1, degDayMax1)

  Calculates the snow parameters.
- subroutine iper_thres_runoff (param, sealedThresh1)

  sets the impervious layer threshold parameter for runoff generation
- subroutine karstic_layer (param, geoUnit0, mask0, SMs_FC0, KsVar_V0, Id0, n_subcells1, upper_bound1, lower_bound1, left_bound1, right_bound1, karstLoss1, L1_Kp)

  calculates the Karstic percolation loss
- subroutine, public canopy_intercept_param (processMatrix, param, LAI0, n_subcells1, upper_bound1, lower_bound1, left_bound1, right_bound1, Id0, mask0, nodata, max_intercept1)

  estimate effective maximum interception capacity at L1
- subroutine aerodynamical_resistance (LAI0, LCover0, param, mask0, Id0, n_subcells1, upper_bound1, lower_bound1, left_bound1, right_bound1, aerodyn_resistance1)

  Regionalization of aerodynamic resistance.

16.64.1 Detailed Description

Multiscale parameter regionalization (MPR).

This module provides the routines for multiscale parameter regionalization (MPR).

Authors

Stephan Thober, Rohini Kumar

Date

Dec 2012

16.64.2 Function/Subroutine Documentation

16.64.2.1 aerodynamical_resistance()

```
subroutine mo_multi_param_reg::aerodynamical_resistance (
             real(dp), dimension(:, :), intent(in) LAI0,
             integer(i4), dimension(:), intent(in) LCover0,
             real(dp), dimension(6), intent(in) param,
             logical, dimension(:, :), intent(in) mask0,
             integer(i4), dimension(:), intent(in) Id0,
             integer(i4), dimension(:), intent(in) n_subcells1,
             integer(i4), dimension(:), intent(in) upper_bound1,
             integer(i4), dimension(:), intent(in) lower_bound1,
             integer(i4), dimension(:), intent(in) left_bound1,
             integer(i4), dimension(:), intent(in) right_bound1,
             real(dp), dimension(:, :), intent(out) aerodyn_resistance1 )
```

Regionalization of aerodynamic resistance.

estimation of aerodynamical resistance Global parameters needed (see mhm_parameter.nml):

- param(1) = canopyheigth_forest
- param(2) = canopyheigth_impervious
- param(3) = canopyheigth_pervious
- param(4) = displacementheight_coeff
- param(5) = roughnesslength_momentum_coeff
- param(6) = roughnesslength_heat_coeff

Parameters

in	*real(dp), dimension(:, :) :: LAI0*	LAI at level-0
in	*integer(i4), dimension(:) :: LCover0*	land cover field
in	*real(dp), dimension(6) :: param*	input parameter
in	*logical, dimension(:, :) :: mask0*	mask at level 0
in	*integer(i4), dimension(:) :: Id0*	Cell ids of hi res field
in	*integer(i4), dimension(:) :: n_subcells1*	number of l0 cells within a l1 cell
in	*integer(i4), dimension(:) :: upper_bound1*	upper row of a l1 cell in l0 grid
in	*integer(i4), dimension(:) :: lower_bound1*	lower row of a l1 cell in l0 grid
in	*integer(i4), dimension(:) :: left_bound1*	left col of a l1 cell in l0 grid
in	*integer(i4), dimension(:) :: right_bound1*	right col of a l1 cell in l0 grid
out	*real(dp), dimension(:, :) :: aerodyn_resistance1*	aerodynmaical resistance

Authors

Matthias Zink

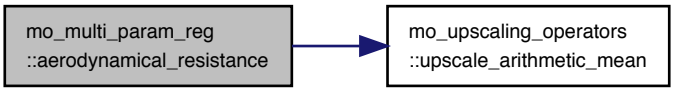
Date

Apr 2013

References mo_common_constants::eps_dp, mo_mpr_constants::karman, mo_upscaling_operators::upscale_↩ arithmetic_mean(), and mo_mpr_constants::windmeasheight.

Referenced by mpr().

Here is the call graph for this function:



Here is the caller graph for this function:

16.64.2.2 baseflow_param()

```
subroutine mo_multi_param_reg::baseflow_param (
             real(dp), dimension(:), intent(in) param,
             integer(i4), dimension(:), intent(in) geoUnit0,
             real(dp), dimension(:), intent(out) k2_0 )
```

baseflow recession parameter

This subroutine calculates the baseflow recession parameter based on the geological units at the Level 0 scale. For each level 0 cell, it assigns the value specified in the parameter array param for the geological unit in this cell. Global parameters needed (see mhm_parameter.nml):

- param(1) = GeoParam(1,:)
- param(2) = GeoParam(2,:)
- ...

Parameters

in	*real(dp), dimension(:) :: param*	list of required parameters
in	*integer(i4), dimension(:) :: geoUnit0*	ids of geological units at L0
out	*real(dp), dimension(:) :: k2_0*	- baseflow recession parameter at Level 0

Authors

Stephan Thober, Rohini Kumar

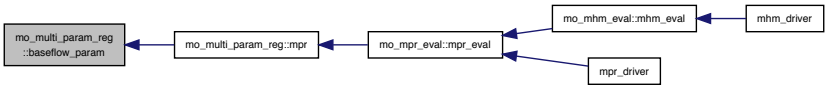
Date

Dec 2012

References mo_mpr_global_variables::geounitlist, and mo_common_constants::nodata_dp.

Referenced by mpr().

Here is the caller graph for this function:



16.64.2.3 canopy_intercept_param()

```
subroutine, public mo_multi_param_reg::canopy_intercept_param (
             integer(i4), dimension(:, :), intent(in) processMatrix,
             real(dp), dimension(:), intent(in) param,
             real(dp), dimension(:, :), intent(in) LAI0,
             integer(i4), dimension(:), intent(in) n_subcells1,
             integer(i4), dimension(:), intent(in) upper_bound1,
             integer(i4), dimension(:), intent(in) lower_bound1,
             integer(i4), dimension(:), intent(in) left_bound1,
             integer(i4), dimension(:), intent(in) right_bound1,
             integer(i4), dimension(:), intent(in) Id0,
             logical, dimension(:, :), intent(in) mask0,
             real(dp), intent(in) nodata,
             real(dp), dimension(:, :), intent(out) max_intercept1 )
```

estimate effective maximum interception capacity at L1

estimate effective maximum interception capacity at L1 for a given Leaf Area Index field. Global parameters needed (see mhm_parameter.nml): Process Case 1:

- param(1) = canopyInterceptionFactor

Parameters

in	*integer(i4), dimension(:, :) :: processMatrix*	indicate processes
in	*real(dp), dimension(:) :: param*	array of global parameters
in	*real(dp), dimension(:, :) :: LAI0*	LAI at level-0(nCells0, time)
in	*integer(i4), dimension(:) :: n_subcells1*	Number of L0 cells within a L1 cell
in	*integer(i4), dimension(:) :: upper_bound1*	Upper row of high resolution block
in	*integer(i4), dimension(:) :: lower_bound1*	Lower row of high resolution block
in	*integer(i4), dimension(:) :: left_bound1*	Left column of high resolution block
in	*integer(i4), dimension(:) :: right_bound1*	Right column of high resolution block
in	*integer(i4), dimension(:) :: Id0*	Cell ids at level 0
in	*logical, dimension(:, :) :: mask0*	mask at level 0 field
in	*real(dp) :: nodata*	- nodata value
out	*real(dp), dimension(:, :) :: max_intercept1*	max interception at level-1(nCells1, time)

Authors

Rohini Kumar

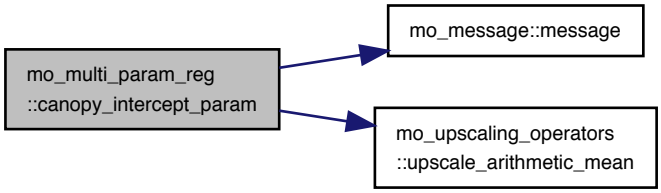
Date

Aug. 2013

References mo_message::message(), and mo_upscaling_operators::upscale_arithmetic_mean().

Referenced by mpr().

Here is the call graph for this function:



Here is the caller graph for this function:

16.64.2.4 iper_thres_runoff()

```
subroutine mo_multi_param_reg::iper_thres_runoff (
            real(dp), dimension(1), intent(in) param,
            real(dp), dimension(:, :, :), intent(out) sealedThresh1 )  [private]
```

sets the impervious layer threshold parameter for runoff generation

to be done by Kumar .... Global parameters needed (see mhm_parameter.nml):

- param(1) = imperviousStorageCapacity

Parameters

in	*real(dp), dimension(1) :: param*	- given threshold parameter
out	*real(dp), dimension(:, :, :) :: sealedThresh1*	

Authors

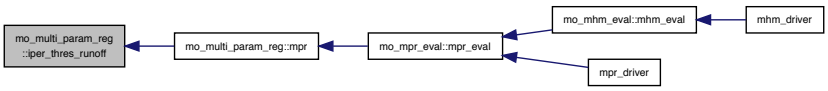
Stephan Thober, Rohini Kumar

Date

Dec 2012

Referenced by mpr().

Here is the caller graph for this function:



16.64.2.5 karstic_layer()

```
subroutine mo_multi_param_reg::karstic_layer (
             real(dp), dimension(3), intent(in) param,
             integer(i4), dimension(:), intent(in) geoUnit0,
             logical, dimension(:, :), intent(in) mask0,
             real(dp), dimension(:), intent(in) SMs_FC0,
             real(dp), dimension(:), intent(in) KsVar_V0,
             integer(i4), dimension(:), intent(in) Id0,
             integer(i4), dimension(:), intent(in) n_subcells1,
             integer(i4), dimension(:), intent(in) upper_bound1,
             integer(i4), dimension(:), intent(in) lower_bound1,
             integer(i4), dimension(:), intent(in) left_bound1,
             integer(i4), dimension(:), intent(in) right_bound1,
             real(dp), dimension(:), intent(out) karstLoss1,
             real(dp), dimension(:), intent(out) L1_Kp )  [private]
```

calculates the Karstic percolation loss

This subroutine calls first the karstic_fraction upscaling routine for determine the karstic fraction area for every Level 1 cell. Then, the karstic percolation loss is estimated given two shape parameters by

$$karstLoss1 = 1 + (fKarArea * param(1)) * ((-1) * *INT(param(2), i4))$$

where $karstLoss1$ is the karstic percolation loss and $fKarArea$ is the fraction of karstic area at level 1 Global parameters needed (see mhm_parameter.nml):

- param(1) = rechargeCoefficient
- param(2) = rechargeFactor_karstic
- param(3) = gain_loss_GWreservoir_karstic

Parameters

in	*real(dp), dimension(3) :: param*	parameters
in	*integer(i4), dimension(:) :: geoUnit0*	id of the Karstic formation
in	*logical, dimension(:, :) :: mask0*	mask at level 0

Parameters

in	*real(dp), dimension(:) :: SMs_FC0*	[-] soil mositure deficit from field capacity w.r.t to saturation
in	*real(dp), dimension(:) :: KsVar_V0*	[-] relative variability of saturated
in	*integer(i4), dimension(:) :: Id0*	Cell ids of hi res field
in	*integer(i4), dimension(:) :: n_subcells1*	number of l0 cells within a l1 cell
in	*integer(i4), dimension(:) :: upper_bound1*	upper row of a l1 cell in l0 grid
in	*integer(i4), dimension(:) :: lower_bound1*	lower row of a l1 cell in l0 grid
in	*integer(i4), dimension(:) :: left_bound1*	left col of a l1 cell in l0 grid
in	*integer(i4), dimension(:) :: right_bound1*	right col of a l1 cell in l0 grid
out	*real(dp), dimension(:) :: karstLoss1*	[-] Karstic percolation loss
out	*real(dp), dimension(:) :: L1_Kp*	[d-1] percolation coefficient

Authors

Rohini Kumar, Stephan Thober

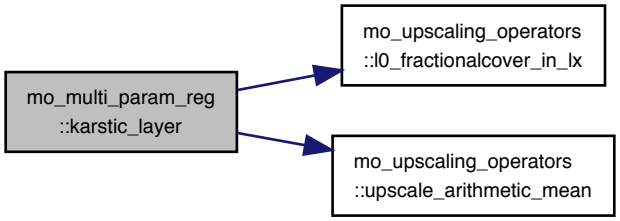
Date

Feb 2013

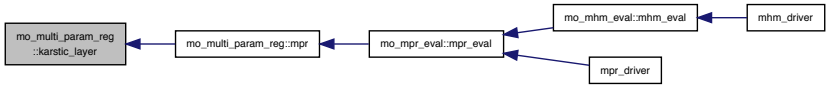
References mo_mpr_global_variables::geounitkar, mo_mpr_global_variables::geounitlist, mo_upscaling_↩ operators::l0_fractionalcover_in_lx(), mo_common_constants::nodata_dp, mo_common_constants::nodata_i4, and mo_upscaling_operators::upscale_arithmetic_mean().

Referenced by mpr().

Here is the call graph for this function:



Here is the caller graph for this function:



16.64.2.6 mpr()

```
subroutine, public mo_multi_param_reg::mpr (
             logical, dimension(:, :), intent(in) mask0,
             integer(i4), dimension(:), intent(in) geoUnit0,
             integer(i4), dimension(:, :), intent(in) soilId0,
             real(dp), dimension(:), intent(in) Asp0,
             real(dp), dimension(:, :), intent(in) gridded_LAI0,
             integer(i4), dimension(:, :), intent(in) LCover0,
             real(dp), dimension(:), intent(in) slope_emp0,
             real(dp), dimension(:), intent(in) y0,
             integer(i4), dimension(:), intent(in) Id0,
             integer(i4), dimension(:), intent(in) upper_bound1,
             integer(i4), dimension(:), intent(in) lower_bound1,
             integer(i4), dimension(:), intent(in) left_bound1,
             integer(i4), dimension(:), intent(in) right_bound1,
             integer(i4), dimension(:), intent(in) n_subcells1,
             real(dp), dimension(:, :, :), intent(inout) fSealed1,
             real(dp), dimension(:, :, :), intent(inout) alpha1,
             real(dp), dimension(:, :, :), intent(inout) degDayInc1,
             real(dp), dimension(:, :, :), intent(inout) degDayMax1,
             real(dp), dimension(:, :, :), intent(inout) degDayNoPre1,
             real(dp), dimension(:, :, :), intent(inout) fAsp1,
             real(dp), dimension(:, :, :), intent(inout) HarSamCoeff1,
             real(dp), dimension(:, :, :), intent(inout) PrieTayAlpha1,
             real(dp), dimension(:, :, :), intent(inout) aeroResist1,
             real(dp), dimension(:, :, :), intent(inout) surfResist1,
             real(dp), dimension(:, :, :), intent(inout) fRoots1,
             real(dp), dimension(:, :, :), intent(inout) kFastFlow1,
             real(dp), dimension(:, :, :), intent(inout) kSlowFlow1,
             real(dp), dimension(:, :, :), intent(inout) kBaseFlow1,
             real(dp), dimension(:, :, :), intent(inout) kPerco1,
             real(dp), dimension(:, :, :), intent(inout) karstLoss1,
             real(dp), dimension(:, :, :), intent(inout) soilMoistFC1,
             real(dp), dimension(:, :, :), intent(inout) soilMoistSat1,
             real(dp), dimension(:, :, :), intent(inout) soilMoistExp1,
             real(dp), dimension(:, :, :), intent(inout) jarvis_thresh_c1,
             real(dp), dimension(:, :, :), intent(inout) tempThresh1,
             real(dp), dimension(:, :, :), intent(inout) unsatThresh1,
             real(dp), dimension(:, :, :), intent(inout) sealedThresh1,
             real(dp), dimension(:, :, :), intent(inout) wiltingPoint1,
             real(dp), dimension(:, :, :), intent(inout) maxInter1,
             real(dp), dimension(:, :, :), intent(inout) petLAIcorFactor,
             real(dp), dimension(:), intent(in), optional, target parameterset )
```

Regionalizing and Upscaling process parameters.

calculating process parameters at L0 scale (Regionalization), like:

- Baseflow recession parameter
- Soil moisture parameters
- PET correction for aspect and upscale these parameters to retrieve effective parameters at scale L1. Further parameter regionalizations are done for:
- snow accumulation and melting parameters
- threshold parameter for runoff generation on impervious layer
- karstic percolation loss

- setting up the Regionalized Routing Parameters

Parameters

in	*logical, dimension(:, :) :: mask0*	mask at level 0 field
in	*integer(i4), dimension(:) :: geoUnit0*	L0 geological units
in	*integer(i4), dimension(:, :) :: soilId0*	soil Ids at level 0
in	*real(dp), dimension(:) :: Asp0*	[degree] Aspect at Level 0
in	*real(dp), dimension(:, :) :: gridded_LAI0*	LAI grid at level 0, with dim2 = time
in	*integer(i4), dimension(:, :) :: LCOVER0*	land cover at level 0
in	*real(dp), dimension(:) :: slope_emp0*	Empirical quantiles of slope
in	*real(dp), dimension(:) :: y0*	y0 at level 0
in	*integer(i4), dimension(:) :: Id0*	Cell ids at level 0
in	*integer(i4), dimension(:) :: upper_bound1*	Upper row of hi res block
in	*integer(i4), dimension(:) :: lower_bound1*	Lower row of hi res block
in	*integer(i4), dimension(:) :: left_bound1*	Left column of hi res block
in	*integer(i4), dimension(:) :: right_bound1*	Right column of hi res block
in	*integer(i4), dimension(:) :: n_subcells1*	Number of L0 cells within a L1 cell
in,out	*real(dp), dimension(:, :, :) :: fSealed1*	[1] fraction of sealed area
in,out	*real(dp), dimension(:, :, :) :: alpha1*	[1] Exponent for the upper reservoir
in,out	*real(dp), dimension(:, :, :) :: degDayInc1*	[d-1 degreeC-1] Increase of the Degree-day factor per mm of increase in precipitation
in,out	*real(dp), dimension(:, :, :) :: degDayMax1*	[mm-1 degreeC-1] Maximum Degree-day factor
in,out	*real(dp), dimension(:, :, :) :: degDayNoPre1*	[mm-1 degreeC-1] Degree-day factor with no precipitation
in,out	*real(dp), dimension(:, :, :) :: fAsp1*	[1] PET correction for Aspect at level 1
in,out	*real(dp), dimension(:, :, :) :: HarSamCoeff1*	[1] PET Hargreaves Samani coeff. at level 1
in,out	*real(dp), dimension(:, :, :) :: PrieTayAlpha1*	[1] PET Priestley Taylor coeff. at level 1
in,out	*real(dp), dimension(:, :, :) :: aeroResist1*	[s m-1] PET aerodynamical resitance at level 1
in,out	*real(dp), dimension(:, :, :) :: surfResist1*	[s m-1] PET bulk surface resitance at level 1
in,out	*real(dp), dimension(:, :, :) :: fRoots1*	fraction of roots in soil horizon
in,out	*real(dp), dimension(:, :, :) :: kFastFlow1*	[10^-3 m] Recession coefficient of the upper reservoir, upper outlet
in,out	*real(dp), dimension(:, :, :) :: kSlowFlow1*	[10^-3 m] Recession coefficient of the upper reservoir, lower outlet
in,out	*real(dp), dimension(:, :, :) :: kBaseFlow1*	Level 1 baseflow recession
in,out	*real(dp), dimension(:, :, :) :: kPerco1*	[d-1] percolation coefficient
in,out	*real(dp), dimension(:, :, :) :: karstLoss1*	
in,out	*real(dp), dimension(:, :, :) :: soilMoistFC1*	[10^-3 m] field capacity
in,out	*real(dp), dimension(:, :, :) :: soilMoistSat1*	[10^-3 m] depth of saturated SM
in,out	*real(dp), dimension(:, :, :) :: soilMoistExp1*	Parameter that determines the rel. contribution to SM, upscal. Bulk den.
in,out	*real(dp), dimension(:, :, :) :: jarvis_thresh_c1*	[1] jarvis critical value for norm SWC
in,out	*real(dp), dimension(:, :, :) :: tempThresh1*	[degreeC] threshold temperature for snow rain
in,out	*real(dp), dimension(:, :, :) :: unsatThresh1*	[10^-3 m] Threshhold water depth in upper reservoir (for Runoff contribution)
in,out	*real(dp), dimension(:, :, :) :: sealedThresh1*	threshold parameter
in,out	*real(dp), dimension(:, :, :) :: wiltingPoint1*	[10^-3 m] permanent wilting point

Parameters

in,out	*real(dp), dimension(:, :, :) :: maxInter1*	
in,out	*real(dp), dimension(:, :, :) :: petLAIcorFactor*	
in	*real(dp), dimension(:), optional :: parameterset*	

Authors

Stephan Thober, Rohini Kumar

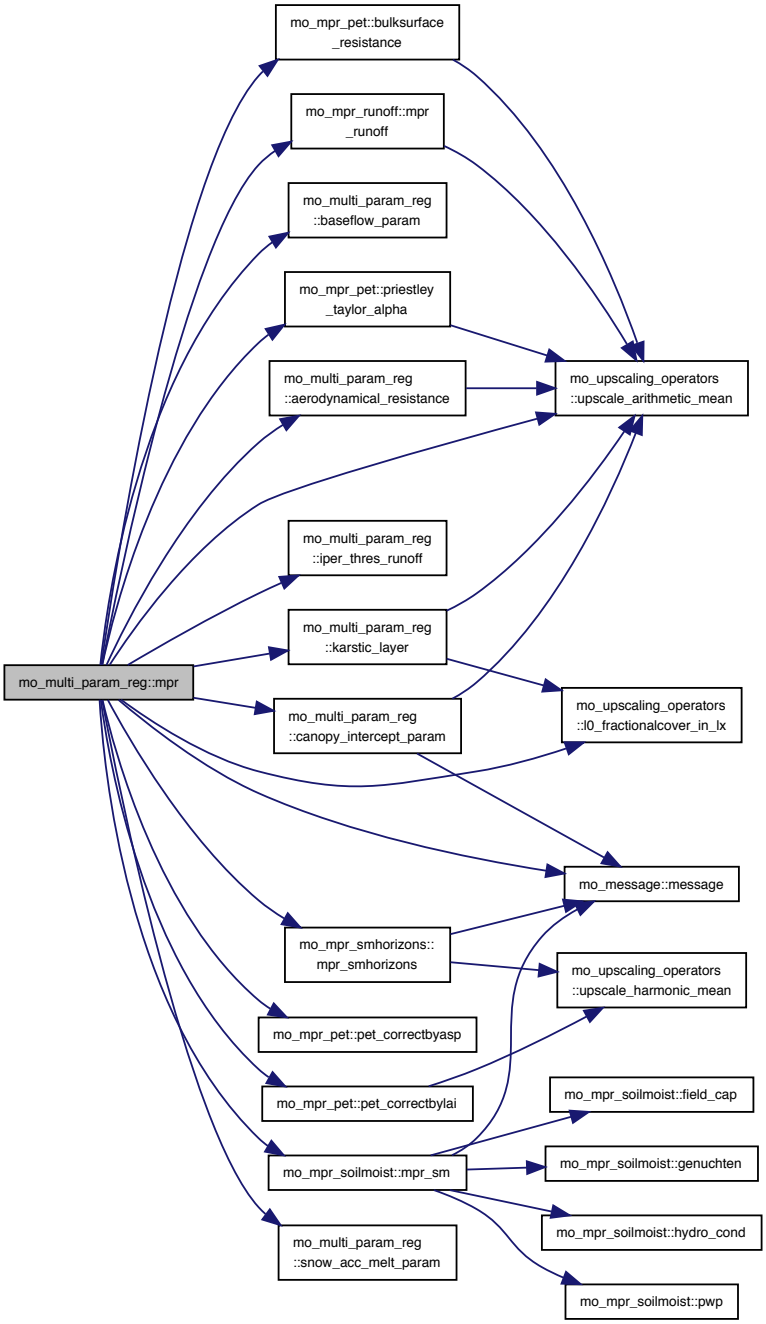
Date

Dec 2012

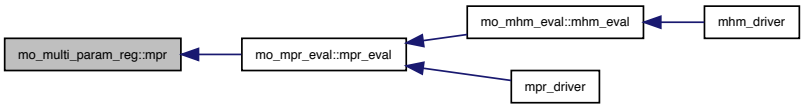
References aerodynamical_resistance(), baseflow_param(), mo_mpr_pet::bulksurface_resistance(), canopy_↩ intercept_param(), mo_mpr_global_variables::fracsealed_cityarea, mo_common_variables::global_parameters, mo_mpr_global_variables::horizondepth_mhm, mo_mpr_global_variables::iflag_soildb, iper_thres_runoff(), karstic_layer(), mo_upscaling_operators::l0_fractionalcover_in_lx(), mo_message::message(), mo_mpr_runoff↩ ::mpr_runoff(), mo_mpr_soilmoist::mpr_sm(), mo_mpr_smhorizons::mpr_smhorizons(), mo_common_constants↩ ::nodata_dp, mo_common_constants::nodata_i4, mo_mpr_global_variables::nsoilhorizons_mhm, mo_mpr_↩ pet::pet_correctbyasp(), mo_mpr_pet::pet_correctbylai(), mo_mpr_pet::priestley_taylor_alpha(), mo_common↩ _variables::processmatrix, snow_acc_melt_param(), mo_mpr_global_variables::soildb, and mo_upscaling_↩ operators::upscale_arithmetic_mean().

Referenced by mo_mpr_eval::mpr_eval().

Here is the call graph for this function:



Here is the caller graph for this function:



16.64.2.7 snow_acc_melt_param()

```
subroutine mo_multi_param_reg::snow_acc_melt_param (
             real(dp), dimension(8), intent(in) param,
             real(dp), dimension(:), intent(in) fForest1,
             real(dp), dimension(:), intent(in) fIperm1,
             real(dp), dimension(:), intent(in) fPerm1,
             real(dp), dimension(:), intent(out) tempThresh1,
             real(dp), dimension(:), intent(out) degDayNoPre1,
             real(dp), dimension(:), intent(out) degDayInc1,
             real(dp), dimension(:), intent(out) degDayMax1 )
```

Calculates the snow parameters.

This subroutine calculates the snow parameters threshold temperature (TT), degree-day factor without precipitation (DD) and maximum degree-day factor (DDmax) as well as increase of degree-day factor per mm of increase in precipitation (IDDP). Global parameters needed (see mhm_parameter.nml):

- param(1) = snowTreshholdTemperature
- param(2) = degreeDayFactor_forest
- param(3) = degreeDayFactor_impervious
- param(4) = degreeDayFactor_pervious
- param(5) = increaseDegreeDayFactorByPrecip
- param(6) = maxDegreeDayFactor_forest
- param(7) = maxDegreeDayFactor_impervious
- param(8) = maxDegreeDayFactor_pervious INTENT(IN)

Parameters

in	*real(dp), dimension(8) :: param*	eight global parameters
in	*real(dp), dimension(:) :: fForest1*	[1] fraction of forest cover
in	*real(dp), dimension(:) :: fIperm1*	[1] fraction of sealed area
in	*real(dp), dimension(:) :: fPerm1*	[1] fraction of permeable area
out	*real(dp), dimension(:) :: tempThresh1*	[degreeC] threshold temperature for snow rain
out	*real(dp), dimension(:) :: degDayNoPre1*	[mm-1 degreeC-1] Degree-day factor with
out	*real(dp), dimension(:) :: degDayInc1*	[d-1 degreeC-1] Increase of the Degree-day
out	*real(dp), dimension(:) :: degDayMax1*	[mm-1 degreeC-1] Maximum Degree-day factor

Authors

Stephan Thober, Rohini Kumar

Date

Dec 2012

Referenced by mpr().

Here is the caller graph for this function:

16.65 mo_ncread Module Reference

Data Types

- interface get_ncvar

Functions/Subroutines

- integer(i4) function, dimension(5), public get_ncdim (Filename, Variable, PrintInfo, ndims)
- subroutine, public get_ncdimatt (Filename, Variable, DimName, DimLen)
- subroutine, public get_ncvaratt (FileName, VarName, AttName, AttValues, fid, dtype)
- subroutine get_ncvar_0d_sp (Filename, VarName, Dat, fid)
- subroutine get_ncvar_0d_dp (Filename, VarName, Dat, fid)
- subroutine get_ncvar_1d_sp (Filename, VarName, Dat, start, a_count, fid)
- subroutine get_ncvar_1d_dp (Filename, VarName, Dat, start, a_count, fid)
- subroutine get_ncvar_2d_sp (Filename, VarName, Dat, start, a_count, fid)
- subroutine get_ncvar_2d_dp (Filename, VarName, Dat, start, a_count, fid)
- subroutine get_ncvar_3d_sp (Filename, VarName, Dat, start, a_count, fid)
- subroutine get_ncvar_3d_dp (Filename, VarName, Dat, start, a_count, fid)
- subroutine get_ncvar_4d_sp (Filename, VarName, Dat, start, a_count, fid)
- subroutine get_ncvar_4d_dp (Filename, VarName, Dat, start, a_count, fid)
- subroutine get_ncvar_5d_sp (Filename, VarName, Dat, start, a_count, fid)
- subroutine get_ncvar_5d_dp (Filename, VarName, Dat, start, a_count, fid)
- subroutine get_ncvar_0d_i4 (Filename, VarName, Dat, fid)
- subroutine get_ncvar_1d_i4 (Filename, VarName, Dat, start, a_count, fid)
- subroutine get_ncvar_2d_i4 (Filename, VarName, Dat, start, a_count, fid)
- subroutine get_ncvar_3d_i4 (Filename, VarName, Dat, start, a_count, fid)
- subroutine get_ncvar_4d_i4 (Filename, VarName, Dat, start, a_count, fid)
- subroutine get_ncvar_5d_i4 (Filename, VarName, Dat, start, a_count, fid)
- subroutine get_ncvar_0d_i1 (Filename, VarName, Dat, fid)
- subroutine get_ncvar_1d_i1 (Filename, VarName, Dat, start, a_count, fid)
- subroutine get_ncvar_2d_i1 (Filename, VarName, Dat, start, a_count, fid)
- subroutine get_ncvar_3d_i1 (Filename, VarName, Dat, start, a_count, fid)
- subroutine get_ncvar_4d_i1 (Filename, VarName, Dat, start, a_count, fid)
- subroutine get_ncvar_5d_i1 (Filename, VarName, Dat, start, a_count, fid)
- integer(i4) function, public ncopen (Fname)

- subroutine, public ncclose (ncid)

- subroutine get_info (Varname, ncid, varid, xtype, dl, Info, ndims)

- subroutine check (status)

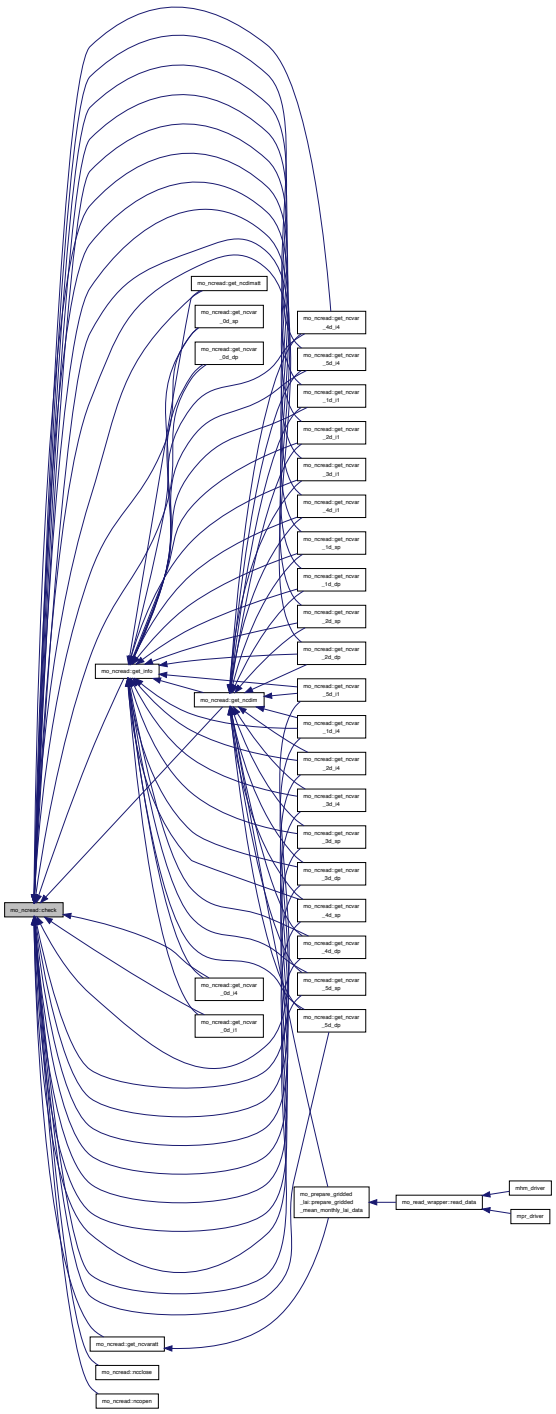
16.65.1 Function/Subroutine Documentation

16.65.1.1 check()

```
subroutine mo_ncread::check (
             integer(i4), intent(in) status )  [private]
```

Referenced by get_info(), get_ncdim(), get_ncdimatt(), get_ncvar_0d_dp(), get_ncvar_0d_i1(), get_ncvar_0d_i4(), get_ncvar_0d_sp(), get_ncvar_1d_dp(), get_ncvar_1d_i1(), get_ncvar_1d_i4(), get_ncvar_1d_sp(), get_ncvar_↩ 2d_dp(), get_ncvar_2d_i1(), get_ncvar_2d_i4(), get_ncvar_2d_sp(), get_ncvar_3d_dp(), get_ncvar_3d_i1(), get↩ _ncvar_3d_i4(), get_ncvar_3d_sp(), get_ncvar_4d_dp(), get_ncvar_4d_i1(), get_ncvar_4d_i4(), get_ncvar_4d_↩ sp(), get_ncvar_5d_dp(), get_ncvar_5d_i1(), get_ncvar_5d_i4(), get_ncvar_5d_sp(), get_ncvaratt(), ncclose(), and ncopen().

Here is the caller graph for this function:



16.65.1.2 get_info()

```
subroutine mo_ncread::get_info (
             character(len = *), intent(in) Varname,
             integer(i4), intent(in) ncid,
```

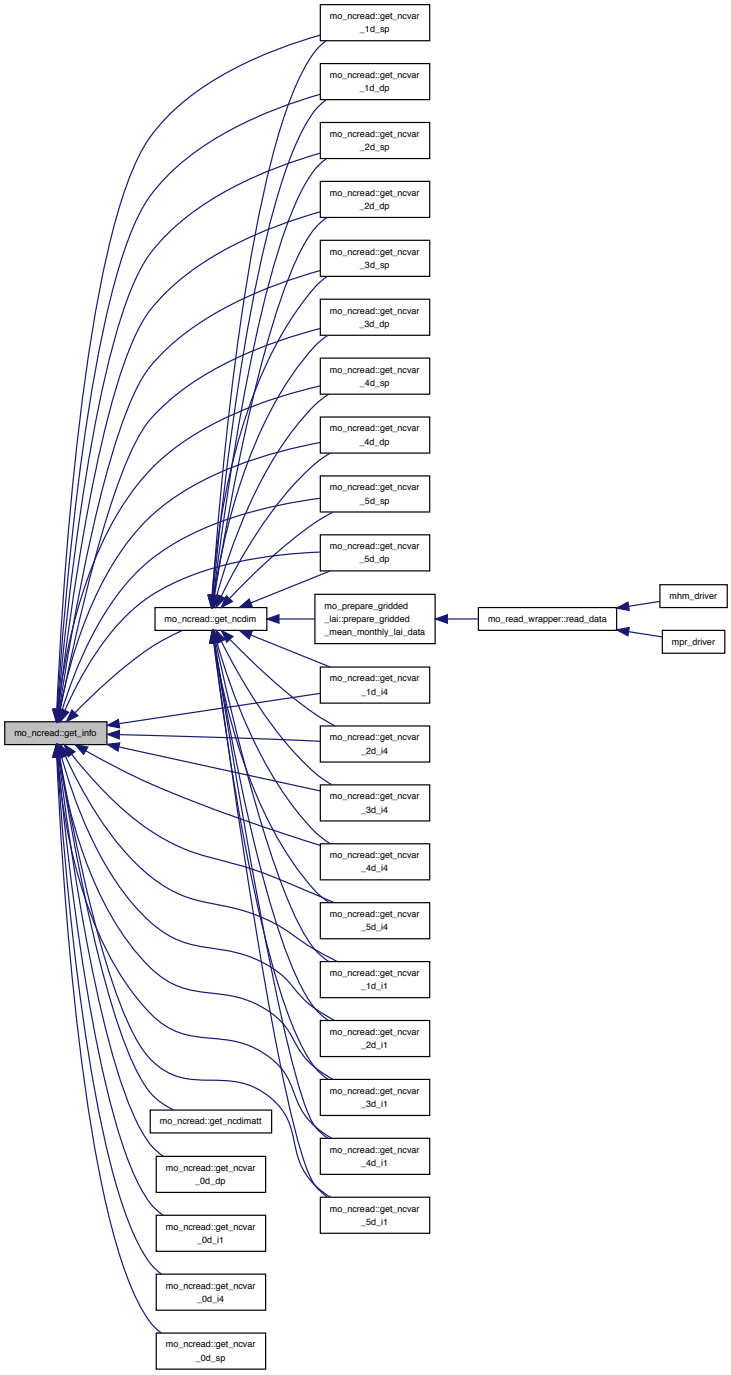
```
integer(i4), intent(out) varid,
integer(i4), intent(out) xtype,
integer(i4), dimension(:), intent(inout), optional dl,
logical, intent(in), optional Info,
integer(i4), intent(out), optional ndims )  [private]
```

References check().

Referenced by get_ncdim(), get_ncdimatt(), get_ncvar_0d_dp(), get_ncvar_0d_i1(), get_ncvar_0d_i4(), get_ncvar↩_0d_sp(), get_ncvar_1d_dp(), get_ncvar_1d_i1(), get_ncvar_1d_i4(), get_ncvar_1d_sp(), get_ncvar_2d_dp(), get↩_ncvar_2d_i1(), get_ncvar_2d_i4(), get_ncvar_2d_sp(), get_ncvar_3d_dp(), get_ncvar_3d_i1(), get_ncvar_3d_i4(), get_ncvar_3d_sp(), get_ncvar_4d_dp(), get_ncvar_4d_i1(), get_ncvar_4d_i4(), get_ncvar_4d_sp(), get_ncvar_5d↩_dp(), get_ncvar_5d_i1(), get_ncvar_5d_i4(), and get_ncvar_5d_sp().

Here is the call graph for this function:

Here is the caller graph for this function:



16.65.1.3 get_ncdim()

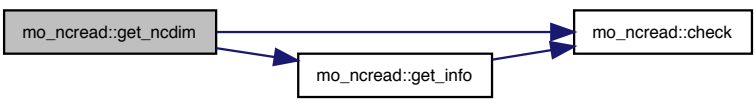
```
integer(i4) function, dimension(5), public mo_ncread::get_ncdim (
             character(len = *), intent(in) Filename,
             character(len = *), intent(in) Variable,
```

```
logical, intent(in), optional PrintInfo,
integer(i4), intent(out), optional ndims )
```

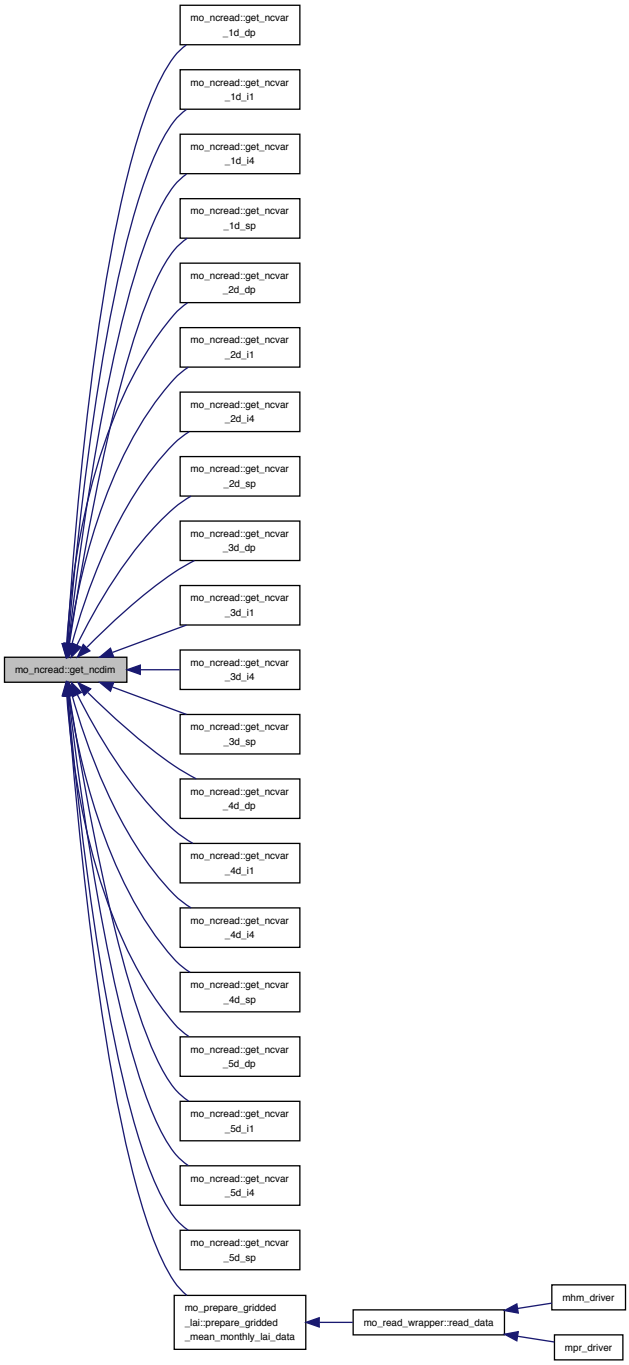
References check(), and get_info().

Referenced by get_ncvar_1d_dp(), get_ncvar_1d_i1(), get_ncvar_1d_i4(), get_ncvar_1d_sp(), get_ncvar_2d_dp(), get_ncvar_2d_i1(), get_ncvar_2d_i4(), get_ncvar_2d_sp(), get_ncvar_3d_dp(), get_ncvar_3d_i1(), get_ncvar_3d_i4(), get_ncvar_3d_sp(), get_ncvar_4d_dp(), get_ncvar_4d_i1(), get_ncvar_4d_i4(), get_ncvar_4d_sp(), get_ncvar_5d_dp(), get_ncvar_5d_i1(), get_ncvar_5d_i4(), get_ncvar_5d_sp(), and mo_prepare_gridded_lai::prepare_gridded_mean_monthly_lai_data().

Here is the call graph for this function:



Here is the caller graph for this function:



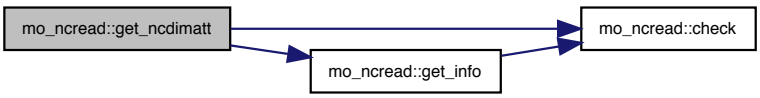
16.65.1.4 get_ncdimatt()

```
subroutine, public mo_ncread::get_ncdimatt (
             character(len = *), intent(in) Filename,
             character(len = *), intent(in) Variable,
```

```
            character(len = *), dimension(:), intent(out), allocatable DimName,
            integer(i4), dimension(:), intent(out), optional, allocatable DimLen )
```

References check(), and get_info().

Here is the call graph for this function:

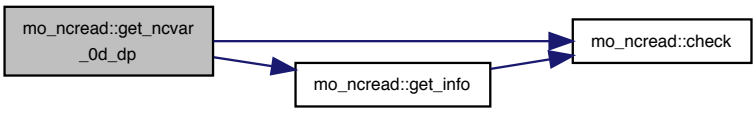


16.65.1.5 get_ncvar_0d_dp()

```
subroutine mo_ncread::get_ncvar_0d_dp (
            character(len = *), intent(in) Filename,
            character(len = *), intent(in) VarName,
            real(dp), intent(inout) Dat,
            integer(i4), intent(in), optional fid )  [private]
```

References check(), and get_info().

Here is the call graph for this function:

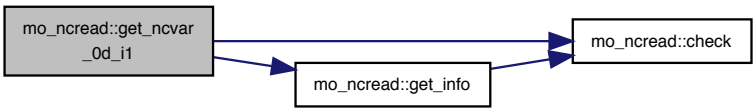


16.65.1.6 get_ncvar_0d_i1()

```
subroutine mo_ncread::get_ncvar_0d_i1 (
            character(len = *), intent(in) Filename,
            character(len = *), intent(in) VarName,
            integer(1), intent(inout) Dat,
            integer(i4), intent(in), optional fid )  [private]
```

References check(), and get_info().

Here is the call graph for this function:

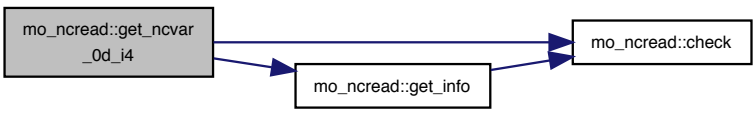


16.65.1.7 get_ncvar_0d_i4()

```
subroutine mo_ncread::get_ncvar_0d_i4 (
             character(len = *), intent(in) Filename,
             character(len = *), intent(in) VarName,
             integer(i4), intent(inout) Dat,
             integer(i4), intent(in), optional fid )  [private]
```

References check(), and get_info().

Here is the call graph for this function:

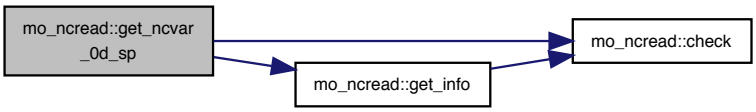


16.65.1.8 get_ncvar_0d_sp()

```
subroutine mo_ncread::get_ncvar_0d_sp (
             character(len = *), intent(in) Filename,
             character(len = *), intent(in) VarName,
             real(sp), intent(inout) Dat,
             integer(i4), intent(in), optional fid )  [private]
```

References check(), and get_info().

Here is the call graph for this function:

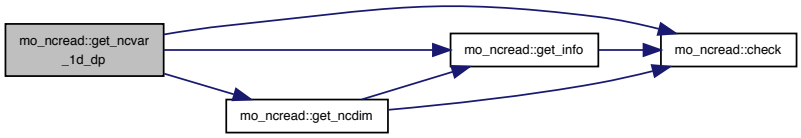


16.65.1.9 get_ncvar_1d_dp()

```
subroutine mo_ncread::get_ncvar_1d_dp (
             character(len = *), intent(in) Filename,
             character(len = *), intent(in) VarName,
             real(dp), dimension(:), intent(inout), allocatable Dat,
             integer(i4), dimension(:), intent(in), optional start,
             integer(i4), dimension(:), intent(in), optional a_count,
             integer(i4), intent(in), optional fid )  [private]
```

References check(), get_info(), and get_ncdim().

Here is the call graph for this function:

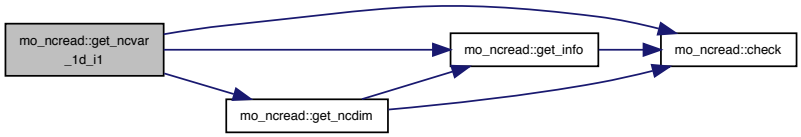


16.65.1.10 get_ncvar_1d_i1()

```
subroutine mo_ncread::get_ncvar_1d_i1 (
             character(len = *), intent(in) Filename,
             character(len = *), intent(in) VarName,
             integer(1), dimension(:), intent(inout), allocatable Dat,
             integer(i4), dimension(:), intent(in), optional start,
             integer(i4), dimension(:), intent(in), optional a_count,
             integer(i4), intent(in), optional fid )  [private]
```

References check(), get_info(), and get_ncdim().

Here is the call graph for this function:

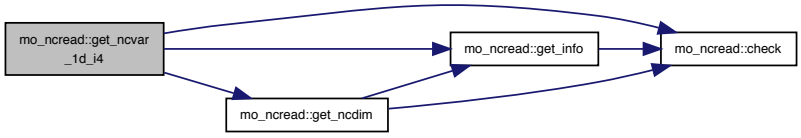


16.65.1.11 get_ncvar_1d_i4()

```
subroutine mo_ncread::get_ncvar_1d_i4 (
             character(len = *), intent(in) Filename,
             character(len = *), intent(in) VarName,
             integer(i4), dimension(:), intent(inout), allocatable Dat,
             integer(i4), dimension(:), intent(in), optional start,
             integer(i4), dimension(:), intent(in), optional a_count,
             integer(i4), intent(in), optional fid )  [private]
```

References check(), get_info(), and get_ncdim().

Here is the call graph for this function:

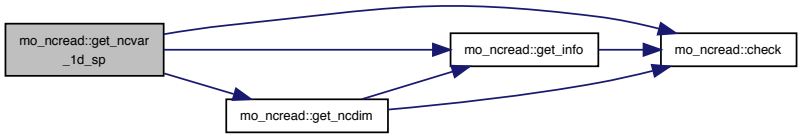


16.65.1.12 get_ncvar_1d_sp()

```
subroutine mo_ncread::get_ncvar_1d_sp (
             character(len = *), intent(in) Filename,
             character(len = *), intent(in) VarName,
             real(sp), dimension(:), intent(inout), allocatable Dat,
             integer(i4), dimension(:), intent(in), optional start,
             integer(i4), dimension(:), intent(in), optional a_count,
             integer(i4), intent(in), optional fid )  [private]
```

References check(), get_info(), and get_ncdim().

Here is the call graph for this function:

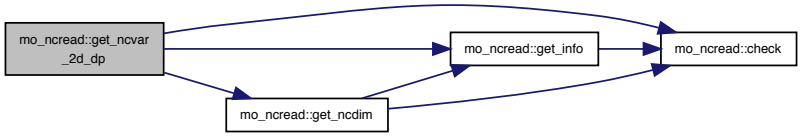


16.65.1.13 get_ncvar_2d_dp()

```
subroutine mo_ncread::get_ncvar_2d_dp (
             character(len = *), intent(in) Filename,
             character(len = *), intent(in) VarName,
             real(dp), dimension(:, :), intent(inout), allocatable Dat,
             integer(i4), dimension(:), intent(in), optional start,
             integer(i4), dimension(:), intent(in), optional a_count,
             integer(i4), intent(in), optional fid )  [private]
```

References check(), get_info(), and get_ncdim().

Here is the call graph for this function:

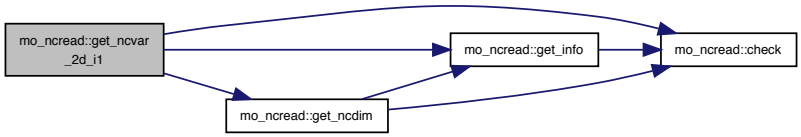


16.65.1.14 get_ncvar_2d_i1()

```
subroutine mo_ncread::get_ncvar_2d_i1 (
             character(len = *), intent(in) Filename,
             character(len = *), intent(in) VarName,
             integer(1), dimension(:, :), intent(inout), allocatable Dat,
             integer(i4), dimension(:), intent(in), optional start,
             integer(i4), dimension(:), intent(in), optional a_count,
             integer(i4), intent(in), optional fid )  [private]
```

References check(), get_info(), and get_ncdim().

Here is the call graph for this function:

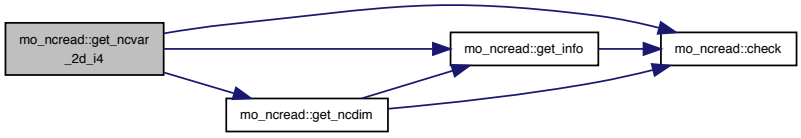


16.65.1.15 get_ncvar_2d_i4()

```
subroutine mo_ncread::get_ncvar_2d_i4 (
            character(len = *), intent(in) Filename,
            character(len = *), intent(in) VarName,
            integer(i4), dimension(:, :), intent(inout), allocatable Dat,
            integer(i4), dimension(:), intent(in), optional start,
            integer(i4), dimension(:), intent(in), optional a_count,
            integer(i4), intent(in), optional fid )  [private]
```

References check(), get_info(), and get_ncdim().

Here is the call graph for this function:

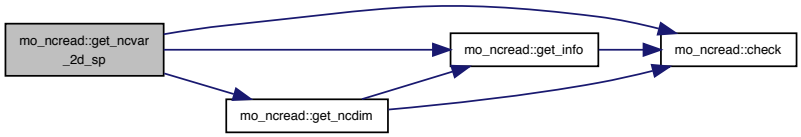


16.65.1.16 get_ncvar_2d_sp()

```
subroutine mo_ncread::get_ncvar_2d_sp (
            character(len = *), intent(in) Filename,
            character(len = *), intent(in) VarName,
            real(sp), dimension(:, :), intent(inout), allocatable Dat,
            integer(i4), dimension(:), intent(in), optional start,
            integer(i4), dimension(:), intent(in), optional a_count,
            integer(i4), intent(in), optional fid )  [private]
```

References check(), get_info(), and get_ncdim().

Here is the call graph for this function:

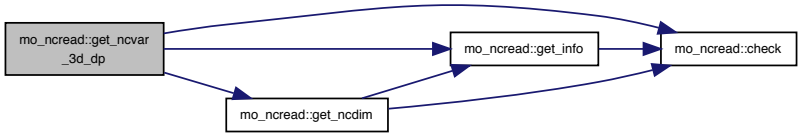


16.65.1.17 get_ncvar_3d_dp()

```
subroutine mo_ncread::get_ncvar_3d_dp (
            character(len = *), intent(in) Filename,
            character(len = *), intent(in) VarName,
            real(dp), dimension(:, :, :), intent(inout), allocatable Dat,
            integer(i4), dimension(:), intent(in), optional start,
            integer(i4), dimension(:), intent(in), optional a_count,
            integer(i4), intent(in), optional fid )  [private]
```

References check(), get_info(), and get_ncdim().

Here is the call graph for this function:

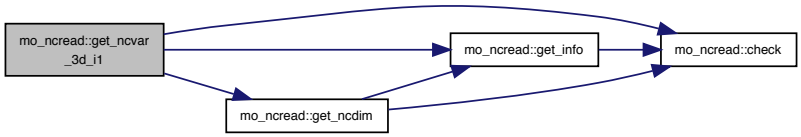


16.65.1.18 get_ncvar_3d_i1()

```
subroutine mo_ncread::get_ncvar_3d_i1 (
            character(len = *), intent(in) Filename,
            character(len = *), intent(in) VarName,
            integer(1), dimension(:, :, :), intent(inout), allocatable Dat,
            integer(i4), dimension(:), intent(in), optional start,
            integer(i4), dimension(:), intent(in), optional a_count,
            integer(i4), intent(in), optional fid )  [private]
```

References check(), get_info(), and get_ncdim().

Here is the call graph for this function:

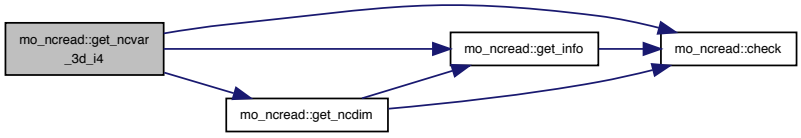


16.65.1.19 get_ncvar_3d_i4()

```
subroutine mo_ncread::get_ncvar_3d_i4 (
            character(len = *), intent(in) Filename,
            character(len = *), intent(in) VarName,
            integer(i4), dimension(:, :, :), intent(inout), allocatable Dat,
            integer(i4), dimension(:), intent(in), optional start,
            integer(i4), dimension(:), intent(in), optional a_count,
            integer(i4), intent(in), optional fid )  [private]
```

References check(), get_info(), and get_ncdim().

Here is the call graph for this function:

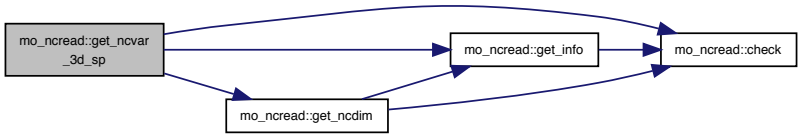


16.65.1.20 get_ncvar_3d_sp()

```
subroutine mo_ncread::get_ncvar_3d_sp (
            character(len = *), intent(in) Filename,
            character(len = *), intent(in) VarName,
            real(sp), dimension(:, :, :), intent(inout), allocatable Dat,
            integer(i4), dimension(:), intent(in), optional start,
            integer(i4), dimension(:), intent(in), optional a_count,
            integer(i4), intent(in), optional fid )  [private]
```

References check(), get_info(), and get_ncdim().

Here is the call graph for this function:

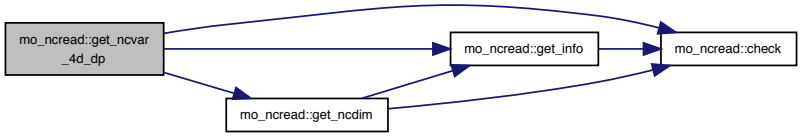


16.65.1.21 get_ncvar_4d_dp()

```
subroutine mo_ncread::get_ncvar_4d_dp (
            character(len = *), intent(in) Filename,
            character(len = *), intent(in) VarName,
            real(dp), dimension(:, :, :, :), intent(inout), allocatable Dat,
            integer(i4), dimension(:), intent(in), optional start,
            integer(i4), dimension(:), intent(in), optional a_count,
            integer(i4), intent(in), optional fid )  [private]
```

References check(), get_info(), and get_ncdim().

Here is the call graph for this function:

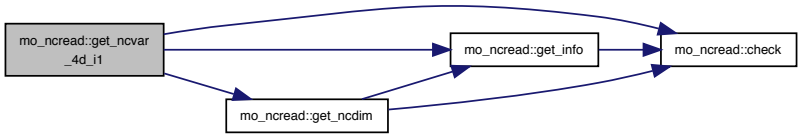


16.65.1.22 get_ncvar_4d_i1()

```
subroutine mo_ncread::get_ncvar_4d_i1 (
            character(len = *), intent(in) Filename,
            character(len = *), intent(in) VarName,
            integer(1), dimension(:, :, :, :), intent(inout), allocatable Dat,
            integer(i4), dimension(:), intent(in), optional start,
            integer(i4), dimension(:), intent(in), optional a_count,
            integer(i4), intent(in), optional fid )  [private]
```

References check(), get_info(), and get_ncdim().

Here is the call graph for this function:

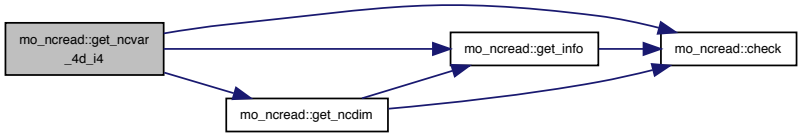


16.65.1.23 get_ncvar_4d_i4()

```
subroutine mo_ncread::get_ncvar_4d_i4 (
            character(len = *), intent(in) Filename,
            character(len = *), intent(in) VarName,
            integer(i4), dimension(:, :, :, :), intent(inout), allocatable Dat,
            integer(i4), dimension(:), intent(in), optional start,
            integer(i4), dimension(:), intent(in), optional a_count,
            integer(i4), intent(in), optional fid )  [private]
```

References check(), get_info(), and get_ncdim().

Here is the call graph for this function:

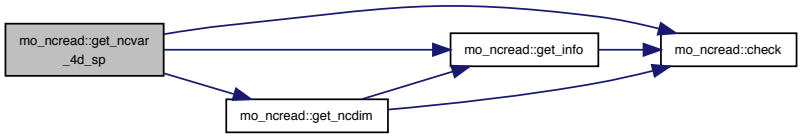


16.65.1.24 get_ncvar_4d_sp()

```
subroutine mo_ncread::get_ncvar_4d_sp (
            character(len = *), intent(in) Filename,
            character(len = *), intent(in) VarName,
            real(sp), dimension(:, :, :, :), intent(inout), allocatable Dat,
            integer(i4), dimension(:), intent(in), optional start,
            integer(i4), dimension(:), intent(in), optional a_count,
            integer(i4), intent(in), optional fid )  [private]
```

References check(), get_info(), and get_ncdim().

Here is the call graph for this function:

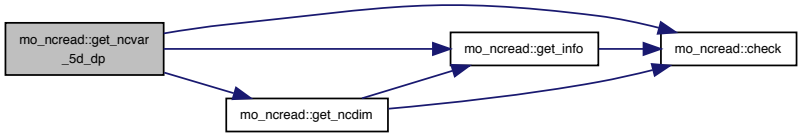


16.65.1.25 get_ncvar_5d_dp()

```
subroutine mo_ncread::get_ncvar_5d_dp (
            character(len = *), intent(in) Filename,
            character(len = *), intent(in) VarName,
            real(dp), dimension(:, :, :, :, :), intent(inout), allocatable Dat,
            integer(i4), dimension(:), intent(in), optional start,
            integer(i4), dimension(:), intent(in), optional a_count,
            integer(i4), intent(in), optional fid )  [private]
```

References check(), get_info(), and get_ncdim().

Here is the call graph for this function:

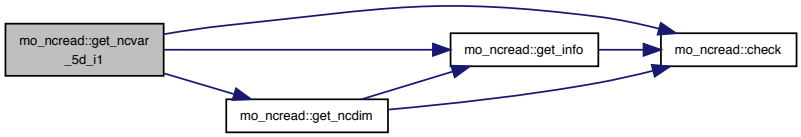


16.65.1.26 get_ncvar_5d_i1()

```
subroutine mo_ncread::get_ncvar_5d_i1 (
            character(len = *), intent(in) Filename,
            character(len = *), intent(in) VarName,
            integer(1), dimension(:, :, :, :, :), intent(inout), allocatable Dat,
            integer(i4), dimension(:), intent(in), optional start,
            integer(i4), dimension(:), intent(in), optional a_count,
            integer(i4), intent(in), optional fid )  [private]
```

References check(), get_info(), and get_ncdim().

Here is the call graph for this function:

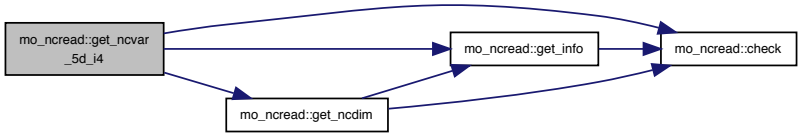


16.65.1.27 get_ncvar_5d_i4()

```
subroutine mo_ncread::get_ncvar_5d_i4 (
            character(len = *), intent(in) Filename,
            character(len = *), intent(in) VarName,
            integer(i4), dimension(:, :, :, :, :), intent(inout), allocatable Dat,
            integer(i4), dimension(:), intent(in), optional start,
            integer(i4), dimension(:), intent(in), optional a_count,
            integer(i4), intent(in), optional fid )  [private]
```

References check(), get_info(), and get_ncdim().

Here is the call graph for this function:

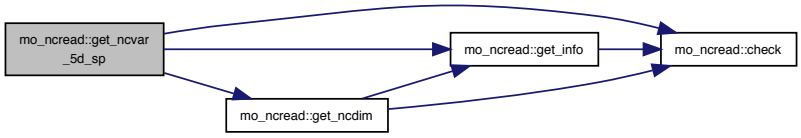


16.65.1.28 get_ncvar_5d_sp()

```
subroutine mo_ncread::get_ncvar_5d_sp (
            character(len = *), intent(in) Filename,
            character(len = *), intent(in) VarName,
            real(sp), dimension(:, :, :, :, :), intent(inout), allocatable Dat,
            integer(i4), dimension(:), intent(in), optional start,
            integer(i4), dimension(:), intent(in), optional a_count,
            integer(i4), intent(in), optional fid )  [private]
```

References check(), get_info(), and get_ncdim().

Here is the call graph for this function:



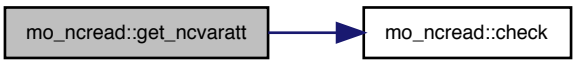
16.65.1.29 get_ncvaratt()

```
subroutine, public mo_ncread::get_ncvaratt (
             character(len = *), intent(in) FileName,
             character(len = *), intent(in) VarName,
             character(len = *), intent(in) AttName,
             character(len = *), intent(out) AttValues,
             integer(i4), intent(in), optional fid,
             integer(i4), intent(out), optional dtype )
```

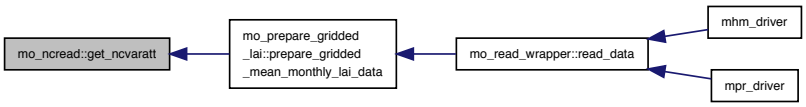
References check().

Referenced by mo_prepare_gridded_lai::prepare_gridded_mean_monthly_lai_data().

Here is the call graph for this function:



Here is the caller graph for this function:



16.65.1.30 ncclose()

```
subroutine, public mo_ncread::ncclose (
             integer(i4), intent(in) ncid )
```

References check().

Here is the call graph for this function:

16.65.1.31 ncopen()

```
integer(i4) function, public mo_ncread::ncopen (
             character(len = *), intent(in) Fname )
```

References check().

Here is the call graph for this function:

16.66 mo_ncwrite Module Reference

Data Types

- type attribute
- type dims
- interface dump_netcdf
- interface var2nc

  Extended dump_netcdf for multiple variables.
- type variable

Functions/Subroutines

- subroutine, public close_netcdf (ncId)
- subroutine, public create_netcdf (Filename, ncid, lfs, netcdf4, deflate_level)
- subroutine dump_netcdf_1d_sp (filename, arr, append, lfs, netcdf4, deflate_level)
- subroutine dump_netcdf_2d_sp (filename, arr, append, lfs, netcdf4, deflate_level)
- subroutine dump_netcdf_3d_sp (filename, arr, append, lfs, netcdf4, deflate_level)
- subroutine dump_netcdf_4d_sp (filename, arr, append, lfs, netcdf4, deflate_level)

- subroutine dump_netcdf_5d_sp (filename, arr, append, lfs, netcdf4, deflate_level)
- subroutine dump_netcdf_1d_dp (filename, arr, append, lfs, netcdf4, deflate_level)
- subroutine dump_netcdf_2d_dp (filename, arr, append, lfs, netcdf4, deflate_level)
- subroutine dump_netcdf_3d_dp (filename, arr, append, lfs, netcdf4, deflate_level)
- subroutine dump_netcdf_4d_dp (filename, arr, append, lfs, netcdf4, deflate_level)
- subroutine dump_netcdf_5d_dp (filename, arr, append, lfs, netcdf4, deflate_level)
- subroutine dump_netcdf_1d_i4 (filename, arr, append, lfs, netcdf4, deflate_level)
- subroutine dump_netcdf_2d_i4 (filename, arr, append, lfs, netcdf4, deflate_level)
- subroutine dump_netcdf_3d_i4 (filename, arr, append, lfs, netcdf4, deflate_level)
- subroutine dump_netcdf_4d_i4 (filename, arr, append, lfs, netcdf4, deflate_level)
- subroutine dump_netcdf_5d_i4 (filename, arr, append, lfs, netcdf4, deflate_level)
- subroutine var2nc_1d_i4 (f_name, arr, dnames, v_name, dim_unlimited, long_name, units, missing_value, attributes, create, ncid, nrec)
- subroutine var2nc_1d_sp (f_name, arr, dnames, v_name, dim_unlimited, long_name, units, missing_value, attributes, create, ncid, nrec)
- subroutine var2nc_1d_dp (f_name, arr, dnames, v_name, dim_unlimited, long_name, units, missing_value, attributes, create, ncid, nrec)
- subroutine var2nc_2d_i4 (f_name, arr, dnames, v_name, dim_unlimited, long_name, units, missing_value, attributes, create, ncid, nrec)
- subroutine var2nc_2d_sp (f_name, arr, dnames, v_name, dim_unlimited, long_name, units, missing_value, attributes, create, ncid, nrec)
- subroutine var2nc_2d_dp (f_name, arr, dnames, v_name, dim_unlimited, long_name, units, missing_value, attributes, create, ncid, nrec)
- subroutine var2nc_3d_i4 (f_name, arr, dnames, v_name, dim_unlimited, long_name, units, missing_value, attributes, create, ncid, nrec)
- subroutine var2nc_3d_sp (f_name, arr, dnames, v_name, dim_unlimited, long_name, units, missing_value, attributes, create, ncid, nrec)
- subroutine var2nc_3d_dp (f_name, arr, dnames, v_name, dim_unlimited, long_name, units, missing_value, attributes, create, ncid, nrec)
- subroutine var2nc_4d_i4 (f_name, arr, dnames, v_name, dim_unlimited, long_name, units, missing_value, attributes, create, ncid, nrec)
- subroutine var2nc_4d_sp (f_name, arr, dnames, v_name, dim_unlimited, long_name, units, missing_value, attributes, create, ncid, nrec)
- subroutine var2nc_4d_dp (f_name, arr, dnames, v_name, dim_unlimited, long_name, units, missing_value, attributes, create, ncid, nrec)
- subroutine var2nc_5d_i4 (f_name, arr, dnames, v_name, dim_unlimited, long_name, units, missing_value, attributes, create, ncid, nrec)
- subroutine var2nc_5d_sp (f_name, arr, dnames, v_name, dim_unlimited, long_name, units, missing_value, attributes, create, ncid, nrec)
- subroutine var2nc_5d_dp (f_name, arr, dnames, v_name, dim_unlimited, long_name, units, missing_value, attributes, create, ncid, nrec)
- subroutine, public write_dynamic_netcdf (ncId, irec)
- subroutine, public write_static_netcdf (ncId)
- integer(i4) function open_netcdf (f_name, create)
- subroutine check (status)

Variables

- integer(i4), parameter, public nmaxdim = 5
- integer(i4), parameter, public nmaxatt = 20
- integer(i4), parameter, public maxlen = 256
- integer(i4), parameter, public ngatt = 20
- integer(i4), parameter, public nattdim = 2
- integer(i4), public nvars
- integer(i4), public ndims
- type(dims), dimension(:), allocatable, public dnc
- type(variable), dimension(:), allocatable, public v
- type(attribute), dimension(ngatt), public gatt

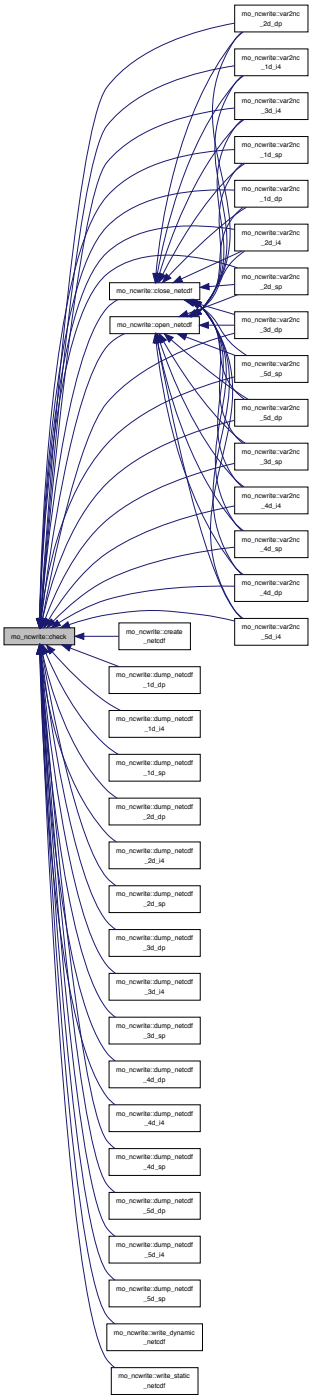
16.66.1 Function/Subroutine Documentation

16.66.1.1 check()

```
subroutine mo_ncwrite::check (
            integer(i4), intent(in) status )  [private]
```

Referenced by close_netcdf(), create_netcdf(), dump_netcdf_1d_dp(), dump_netcdf_1d_i4(), dump_netcdf_1d_↩
sp(), dump_netcdf_2d_dp(), dump_netcdf_2d_i4(), dump_netcdf_2d_sp(), dump_netcdf_3d_dp(), dump_netcdf↩
_3d_i4(), dump_netcdf_3d_sp(), dump_netcdf_4d_dp(), dump_netcdf_4d_i4(), dump_netcdf_4d_sp(), dump_↩
netcdf_5d_dp(), dump_netcdf_5d_i4(), dump_netcdf_5d_sp(), open_netcdf(), var2nc_1d_dp(), var2nc_1d_i4(), var2nc_1d_sp(), var2nc_2d_dp(), var2nc_2d_i4(), var2nc_2d_sp(), var2nc_3d_dp(), var2nc_3d_i4(), var2nc_3d↩
_sp(), var2nc_4d_dp(), var2nc_4d_i4(), var2nc_4d_sp(), var2nc_5d_dp(), var2nc_5d_i4(), var2nc_5d_sp(), write↩
_dynamic_netcdf(), and write_static_netcdf().

Here is the caller graph for this function:



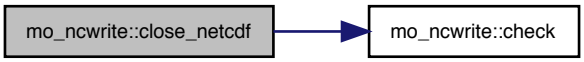
16.66.1.2 close_netcdf()

```
subroutine, public mo_ncwrite::close_netcdf (
            integer(i4), intent(in) ncId )
```

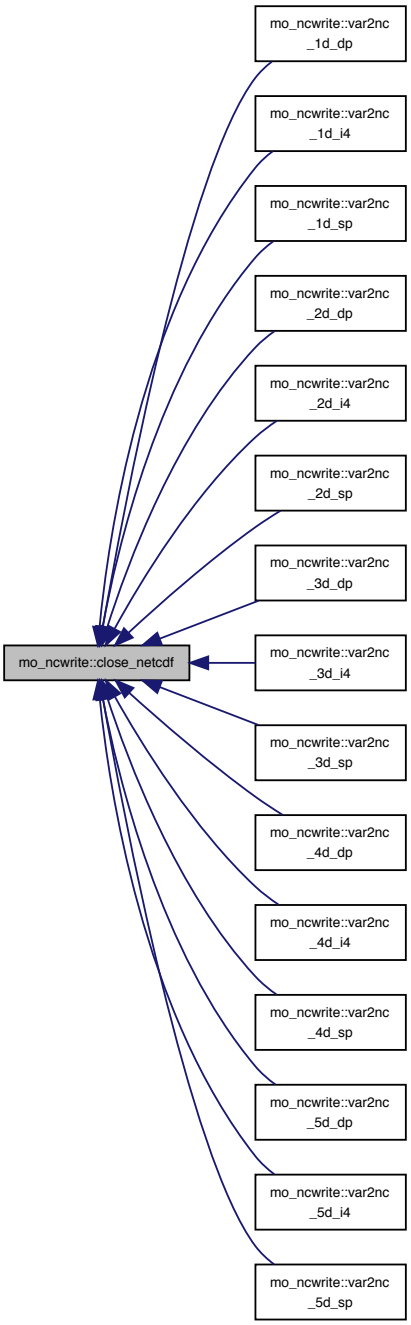
References check().

Referenced by var2nc_1d_dp(), var2nc_1d_i4(), var2nc_1d_sp(), var2nc_2d_dp(), var2nc_2d_i4(), var2nc_2d_sp(), var2nc_3d_dp(), var2nc_3d_i4(), var2nc_3d_sp(), var2nc_4d_dp(), var2nc_4d_i4(), var2nc_4d_sp(), var2nc_5d_↩ dp(), var2nc_5d_i4(), and var2nc_5d_sp().

Here is the call graph for this function:



Here is the caller graph for this function:



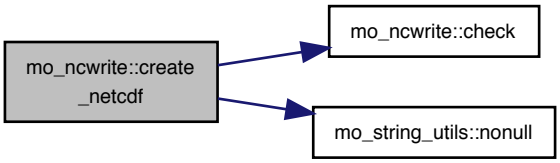
16.66.1.3 create_netcdf()

```
subroutine, public mo_ncwrite::create_netcdf (
             character(len = *), intent(in) Filename,
             integer(i4), intent(out) ncid,
```

```
            logical, intent(in), optional lfs,
            logical, intent(in), optional netcdf4,
            integer(i4), intent(in), optional deflate_level )
```

References check(), dnc, gatt, ndims, ngatt, mo_string_utils::nonull(), nvars, and v.

Here is the call graph for this function:

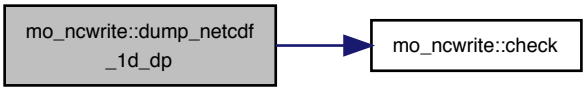


16.66.1.4 dump_netcdf_1d_dp()

```
subroutine mo_ncwrite::dump_netcdf_1d_dp (
            character(len = *), intent(in) filename,
            real(dp), dimension(:), intent(in) arr,
            logical, intent(in), optional append,
            logical, intent(in), optional lfs,
            logical, intent(in), optional netcdf4,
            integer(i4), intent(in), optional deflate_level )  [private]
```

References check(), and ndims.

Here is the call graph for this function:



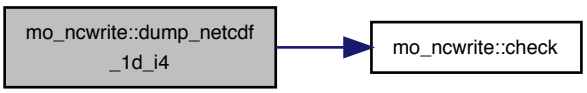
16.66.1.5 dump_netcdf_1d_i4()

```
subroutine mo_ncwrite::dump_netcdf_1d_i4 (
            character(len = *), intent(in) filename,
            integer(i4), dimension(:), intent(in) arr,
            logical, intent(in), optional append,
            logical, intent(in), optional lfs,
```

```
            logical, intent(in), optional netcdf4,
            integer(i4), intent(in), optional deflate_level )  [private]
```

References check(), and ndims.

Here is the call graph for this function:

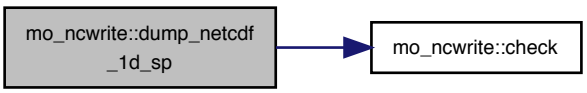


16.66.1.6 dump_netcdf_1d_sp()

```
subroutine mo_ncwrite::dump_netcdf_1d_sp (
            character(len = *), intent(in) filename,
            real(sp), dimension(:), intent(in) arr,
            logical, intent(in), optional append,
            logical, intent(in), optional lfs,
            logical, intent(in), optional netcdf4,
            integer(i4), intent(in), optional deflate_level )  [private]
```

References check(), and ndims.

Here is the call graph for this function:

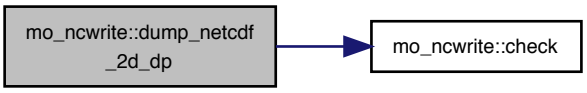


16.66.1.7 dump_netcdf_2d_dp()

```
subroutine mo_ncwrite::dump_netcdf_2d_dp (
            character(len = *), intent(in) filename,
            real(dp), dimension(:, :), intent(in) arr,
            logical, intent(in), optional append,
            logical, intent(in), optional lfs,
            logical, intent(in), optional netcdf4,
            integer(i4), intent(in), optional deflate_level )  [private]
```

References check(), and ndims.

Here is the call graph for this function:

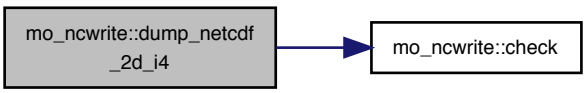


16.66.1.8 dump_netcdf_2d_i4()

```
subroutine mo_ncwrite::dump_netcdf_2d_i4 (
             character(len = *), intent(in) filename,
             integer(i4), dimension(:, :), intent(in) arr,
             logical, intent(in), optional append,
             logical, intent(in), optional lfs,
             logical, intent(in), optional netcdf4,
             integer(i4), intent(in), optional deflate_level )  [private]
```

References check(), and ndims.

Here is the call graph for this function:

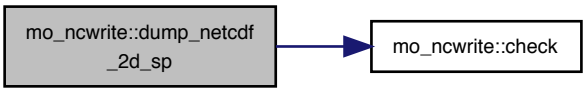


16.66.1.9 dump_netcdf_2d_sp()

```
subroutine mo_ncwrite::dump_netcdf_2d_sp (
             character(len = *), intent(in) filename,
             real(sp), dimension(:, :), intent(in) arr,
             logical, intent(in), optional append,
             logical, intent(in), optional lfs,
             logical, intent(in), optional netcdf4,
             integer(i4), intent(in), optional deflate_level )  [private]
```

References check(), and ndims.

Here is the call graph for this function:

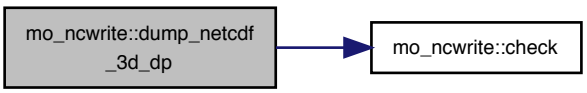


16.66.1.10 dump_netcdf_3d_dp()

```
subroutine mo_ncwrite::dump_netcdf_3d_dp (
             character(len = *), intent(in) filename,
             real(dp), dimension(:, :, :), intent(in) arr,
             logical, intent(in), optional append,
             logical, intent(in), optional lfs,
             logical, intent(in), optional netcdf4,
             integer(i4), intent(in), optional deflate_level )  [private]
```

References check(), and ndims.

Here is the call graph for this function:

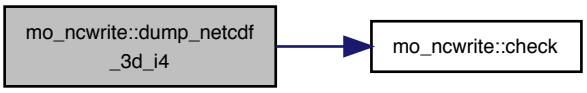


16.66.1.11 dump_netcdf_3d_i4()

```
subroutine mo_ncwrite::dump_netcdf_3d_i4 (
             character(len = *), intent(in) filename,
             integer(i4), dimension(:, :, :), intent(in) arr,
             logical, intent(in), optional append,
             logical, intent(in), optional lfs,
             logical, intent(in), optional netcdf4,
             integer(i4), intent(in), optional deflate_level )  [private]
```

References check(), and ndims.

Here is the call graph for this function:

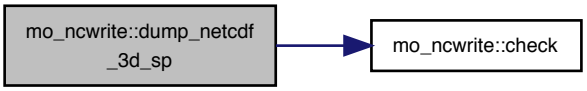


16.66.1.12 dump_netcdf_3d_sp()

```
subroutine mo_ncwrite::dump_netcdf_3d_sp (
             character(len = *), intent(in) filename,
             real(sp), dimension(:, :, :), intent(in) arr,
             logical, intent(in), optional append,
             logical, intent(in), optional lfs,
             logical, intent(in), optional netcdf4,
             integer(i4), intent(in), optional deflate_level )  [private]
```

References check(), and ndims.

Here is the call graph for this function:

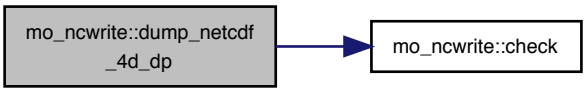


16.66.1.13 dump_netcdf_4d_dp()

```
subroutine mo_ncwrite::dump_netcdf_4d_dp (
             character(len = *), intent(in) filename,
             real(dp), dimension(:, :, :, :), intent(in) arr,
             logical, intent(in), optional append,
             logical, intent(in), optional lfs,
             logical, intent(in), optional netcdf4,
             integer(i4), intent(in), optional deflate_level )  [private]
```

References check(), and ndims.

Here is the call graph for this function:

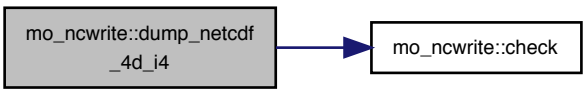


16.66.1.14 dump_netcdf_4d_i4()

```
subroutine mo_ncwrite::dump_netcdf_4d_i4 (
             character(len = *), intent(in) filename,
             integer(i4), dimension(:, :, :, :), intent(in) arr,
             logical, intent(in), optional append,
             logical, intent(in), optional lfs,
             logical, intent(in), optional netcdf4,
             integer(i4), intent(in), optional deflate_level )  [private]
```

References check(), and ndims.

Here is the call graph for this function:

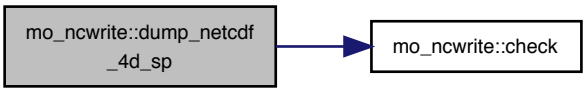


16.66.1.15 dump_netcdf_4d_sp()

```
subroutine mo_ncwrite::dump_netcdf_4d_sp (
             character(len = *), intent(in) filename,
             real(sp), dimension(:, :, :, :), intent(in) arr,
             logical, intent(in), optional append,
             logical, intent(in), optional lfs,
             logical, intent(in), optional netcdf4,
             integer(i4), intent(in), optional deflate_level )  [private]
```

References check(), and ndims.

Here is the call graph for this function:

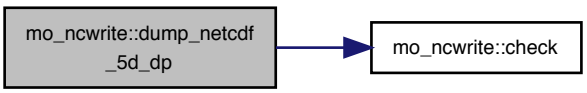


16.66.1.16 dump_netcdf_5d_dp()

```
subroutine mo_ncwrite::dump_netcdf_5d_dp (
             character(len = *), intent(in) filename,
             real(dp), dimension(:, :, :, :, :), intent(in) arr,
             logical, intent(in), optional append,
             logical, intent(in), optional lfs,
             logical, intent(in), optional netcdf4,
             integer(i4), intent(in), optional deflate_level )  [private]
```

References check(), and ndims.

Here is the call graph for this function:

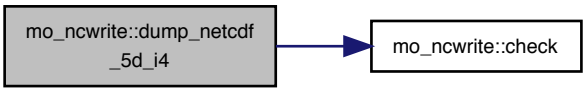


16.66.1.17 dump_netcdf_5d_i4()

```
subroutine mo_ncwrite::dump_netcdf_5d_i4 (
             character(len = *), intent(in) filename,
             integer(i4), dimension(:, :, :, :, :), intent(in) arr,
             logical, intent(in), optional append,
             logical, intent(in), optional lfs,
             logical, intent(in), optional netcdf4,
             integer(i4), intent(in), optional deflate_level )  [private]
```

References check(), and ndims.

Here is the call graph for this function:

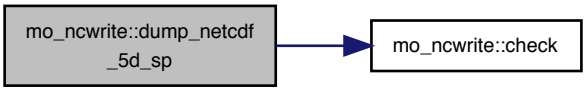


16.66.1.18 dump_netcdf_5d_sp()

```
subroutine mo_ncwrite::dump_netcdf_5d_sp (
             character(len = *), intent(in) filename,
             real(sp), dimension(:, :, :, :, :), intent(in) arr,
             logical, intent(in), optional append,
             logical, intent(in), optional lfs,
             logical, intent(in), optional netcdf4,
             integer(i4), intent(in), optional deflate_level )  [private]
```

References check(), and ndims.

Here is the call graph for this function:



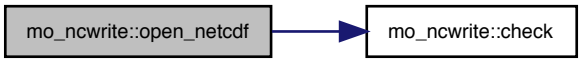
16.66.1.19 open_netcdf()

```
integer(i4) function mo_ncwrite::open_netcdf (
             character(len = *), intent(in) f_name,
             logical, intent(in) create )  [private]
```

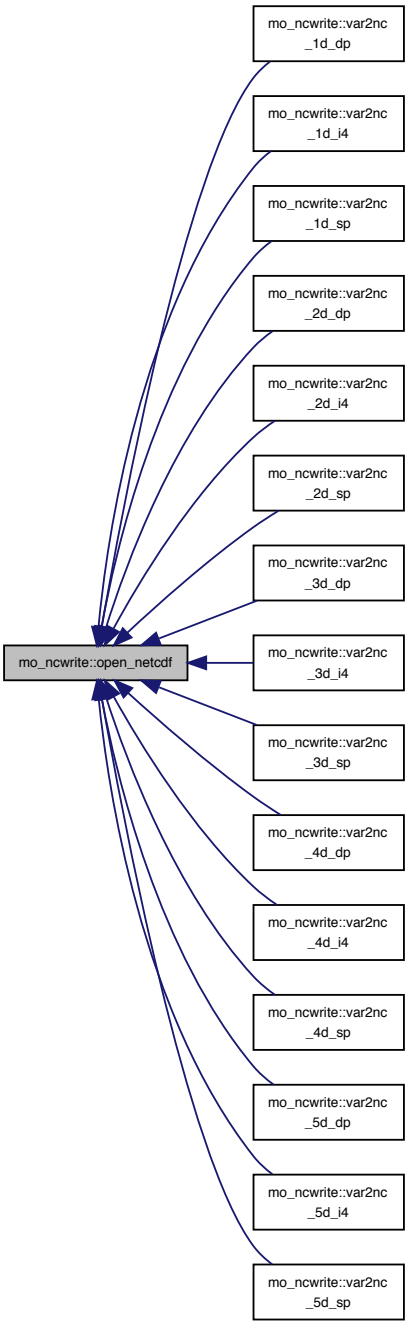
References check().

Referenced by var2nc_1d_dp(), var2nc_1d_i4(), var2nc_1d_sp(), var2nc_2d_dp(), var2nc_2d_i4(), var2nc_2d_sp(), var2nc_3d_dp(), var2nc_3d_i4(), var2nc_3d_sp(), var2nc_4d_dp(), var2nc_4d_i4(), var2nc_4d_sp(), var2nc_5d_↩ dp(), var2nc_5d_i4(), and var2nc_5d_sp().

Here is the call graph for this function:



Here is the caller graph for this function:



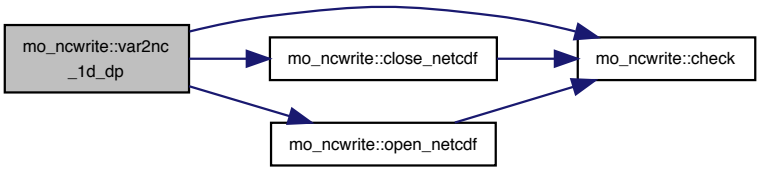
16.66.1.20 var2nc_1d_dp()

```
subroutine mo_ncwrite::var2nc_1d_dp (
            character(len = *), intent(in) f_name,
            real(dp), dimension(:), intent(in) arr,
```

```
            character(len = *), dimension(:), intent(in) dnames,
            character(len = *), intent(in) v_name,
            integer(i4), intent(in), optional dim_unlimited,
            character(len = *), intent(in), optional long_name,
            character(len = *), intent(in), optional units,
            real(dp), intent(in), optional missing_value,
            character(256), dimension(:, :), intent(in), optional attributes,
            logical, intent(in), optional create,
            integer(i4), intent(inout), optional ncid,
            integer(i4), intent(in), optional nrec )  [private]
```

References check(), close_netcdf(), ndims, and open_netcdf().

Here is the call graph for this function:

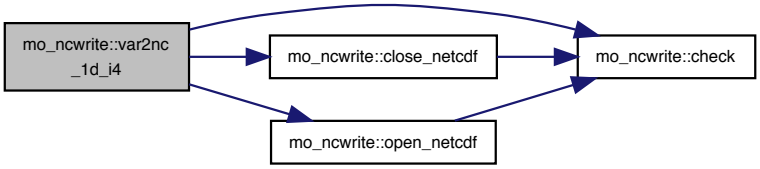


16.66.1.21 var2nc_1d_i4()

```
subroutine mo_ncwrite::var2nc_1d_i4 (
            character(len = *), intent(in) f_name,
            integer(i4), dimension(:), intent(in) arr,
            character(len = *), dimension(:), intent(in) dnames,
            character(len = *), intent(in) v_name,
            integer(i4), intent(in), optional dim_unlimited,
            character(len = *), intent(in), optional long_name,
            character(len = *), intent(in), optional units,
            integer(i4), intent(in), optional missing_value,
            character(256), dimension(:, :), intent(in), optional attributes,
            logical, intent(in), optional create,
            integer(i4), intent(inout), optional ncid,
            integer(i4), intent(in), optional nrec )  [private]
```

References check(), close_netcdf(), ndims, and open_netcdf().

Here is the call graph for this function:

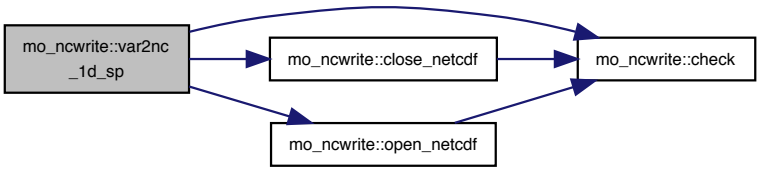


16.66.1.22 var2nc_1d_sp()

```
subroutine mo_ncwrite::var2nc_1d_sp (
             character(len = *), intent(in) f_name,
             real(sp), dimension(:), intent(in) arr,
             character(len = *), dimension(:), intent(in) dnames,
             character(len = *), intent(in) v_name,
             integer(i4), intent(in), optional dim_unlimited,
             character(len = *), intent(in), optional long_name,
             character(len = *), intent(in), optional units,
             real(sp), intent(in), optional missing_value,
             character(256), dimension(:, :), intent(in), optional attributes,
             logical, intent(in), optional create,
             integer(i4), intent(inout), optional ncid,
             integer(i4), intent(in), optional nrec )  [private]
```

References check(), close_netcdf(), ndims, and open_netcdf().

Here is the call graph for this function:



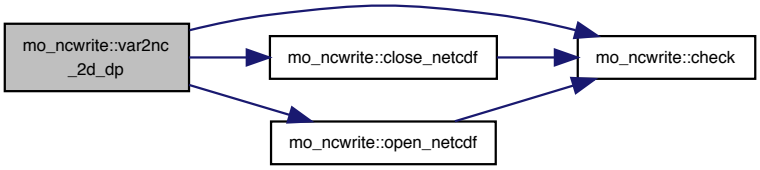
16.66.1.23 var2nc_2d_dp()

```
subroutine mo_ncwrite::var2nc_2d_dp (
             character(len = *), intent(in) f_name,
             real(dp), dimension(:, :), intent(in) arr,
             character(len = *), dimension(:), intent(in) dnames,
```

```
            character(len = *), intent(in) v_name,
            integer(i4), intent(in), optional dim_unlimited,
            character(len = *), intent(in), optional long_name,
            character(len = *), intent(in), optional units,
            real(dp), intent(in), optional missing_value,
            character(256), dimension(:, :), intent(in), optional attributes,
            logical, intent(in), optional create,
            integer(i4), intent(inout), optional ncid,
            integer(i4), intent(in), optional nrec )  [private]
```

References check(), close_netcdf(), ndims, and open_netcdf().

Here is the call graph for this function:

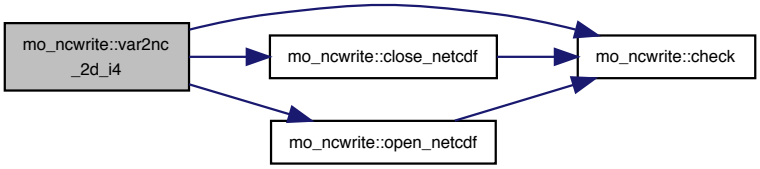


16.66.1.24 var2nc_2d_i4()

```
subroutine mo_ncwrite::var2nc_2d_i4 (
            character(len = *), intent(in) f_name,
            integer(i4), dimension(:, :), intent(in) arr,
            character(len = *), dimension(:), intent(in) dnames,
            character(len = *), intent(in) v_name,
            integer(i4), intent(in), optional dim_unlimited,
            character(len = *), intent(in), optional long_name,
            character(len = *), intent(in), optional units,
            integer(i4), intent(in), optional missing_value,
            character(256), dimension(:, :), intent(in), optional attributes,
            logical, intent(in), optional create,
            integer(i4), intent(inout), optional ncid,
            integer(i4), intent(in), optional nrec )  [private]
```

References check(), close_netcdf(), ndims, and open_netcdf().

Here is the call graph for this function:

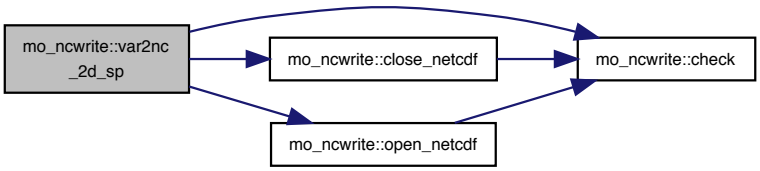


16.66.1.25 var2nc_2d_sp()

```
subroutine mo_ncwrite::var2nc_2d_sp (
             character(len = *), intent(in) f_name,
             real(sp), dimension(:, :), intent(in) arr,
             character(len = *), dimension(:), intent(in) dnames,
             character(len = *), intent(in) v_name,
             integer(i4), intent(in), optional dim_unlimited,
             character(len = *), intent(in), optional long_name,
             character(len = *), intent(in), optional units,
             real(sp), intent(in), optional missing_value,
             character(256), dimension(:, :), intent(in), optional attributes,
             logical, intent(in), optional create,
             integer(i4), intent(inout), optional ncid,
             integer(i4), intent(in), optional nrec )  [private]
```

References check(), close_netcdf(), ndims, and open_netcdf().

Here is the call graph for this function:



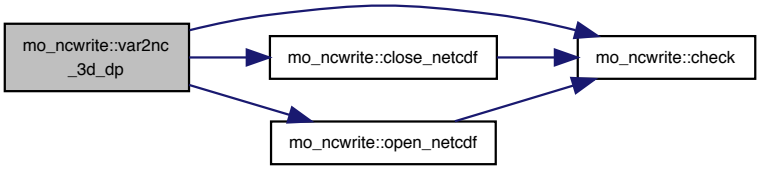
16.66.1.26 var2nc_3d_dp()

```
subroutine mo_ncwrite::var2nc_3d_dp (
             character(len = *), intent(in) f_name,
             real(dp), dimension(:, :, :), intent(in) arr,
             character(len = *), dimension(:), intent(in) dnames,
```

```
            character(len = *), intent(in) v_name,
            integer(i4), intent(in), optional dim_unlimited,
            character(len = *), intent(in), optional long_name,
            character(len = *), intent(in), optional units,
            real(dp), intent(in), optional missing_value,
            character(256), dimension(:, :), intent(in), optional attributes,
            logical, intent(in), optional create,
            integer(i4), intent(inout), optional ncid,
            integer(i4), intent(in), optional nrec )  [private]
```

References check(), close_netcdf(), ndims, and open_netcdf().

Here is the call graph for this function:

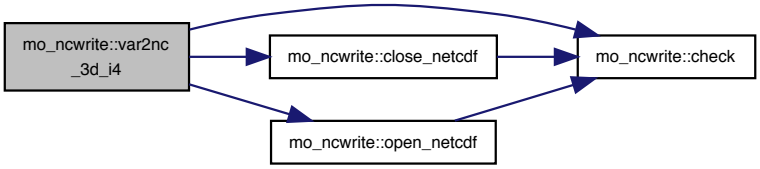


16.66.1.27 var2nc_3d_i4()

```
subroutine mo_ncwrite::var2nc_3d_i4 (
            character(len = *), intent(in) f_name,
            integer(i4), dimension(:, :, :), intent(in) arr,
            character(len = *), dimension(:), intent(in) dnames,
            character(len = *), intent(in) v_name,
            integer(i4), intent(in), optional dim_unlimited,
            character(len = *), intent(in), optional long_name,
            character(len = *), intent(in), optional units,
            integer(i4), intent(in), optional missing_value,
            character(256), dimension(:, :), intent(in), optional attributes,
            logical, intent(in), optional create,
            integer(i4), intent(inout), optional ncid,
            integer(i4), intent(in), optional nrec )  [private]
```

References check(), close_netcdf(), ndims, and open_netcdf().

Here is the call graph for this function:

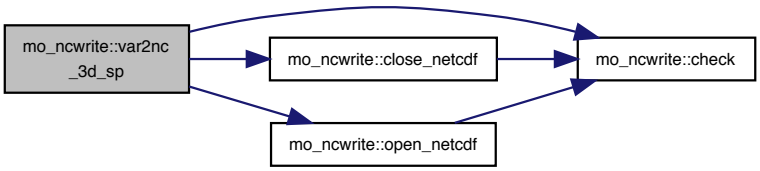


16.66.1.28 var2nc_3d_sp()

```
subroutine mo_ncwrite::var2nc_3d_sp (
             character(len = *), intent(in) f_name,
             real(sp), dimension(:, :, :), intent(in) arr,
             character(len = *), dimension(:), intent(in) dnames,
             character(len = *), intent(in) v_name,
             integer(i4), intent(in), optional dim_unlimited,
             character(len = *), intent(in), optional long_name,
             character(len = *), intent(in), optional units,
             real(sp), intent(in), optional missing_value,
             character(256), dimension(:, :), intent(in), optional attributes,
             logical, intent(in), optional create,
             integer(i4), intent(inout), optional ncid,
             integer(i4), intent(in), optional nrec )  [private]
```

References check(), close_netcdf(), ndims, and open_netcdf().

Here is the call graph for this function:



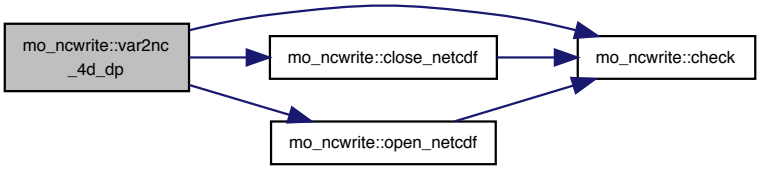
16.66.1.29 var2nc_4d_dp()

```
subroutine mo_ncwrite::var2nc_4d_dp (
             character(len = *), intent(in) f_name,
             real(dp), dimension(:, :, :, :), intent(in) arr,
             character(len = *), dimension(:), intent(in) dnames,
```

```
            character(len = *), intent(in) v_name,
            integer(i4), intent(in), optional dim_unlimited,
            character(len = *), intent(in), optional long_name,
            character(len = *), intent(in), optional units,
            real(dp), intent(in), optional missing_value,
            character(256), dimension(:, :), intent(in), optional attributes,
            logical, intent(in), optional create,
            integer(i4), intent(inout), optional ncid,
            integer(i4), intent(in), optional nrec )  [private]
```

References check(), close_netcdf(), ndims, and open_netcdf().

Here is the call graph for this function:

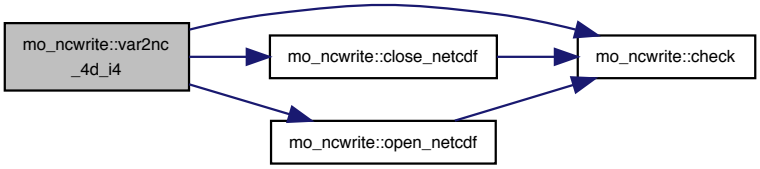


16.66.1.30 var2nc_4d_i4()

```
subroutine mo_ncwrite::var2nc_4d_i4 (
            character(len = *), intent(in) f_name,
            integer(i4), dimension(:, :, :, :), intent(in) arr,
            character(len = *), dimension(:), intent(in) dnames,
            character(len = *), intent(in) v_name,
            integer(i4), intent(in), optional dim_unlimited,
            character(len = *), intent(in), optional long_name,
            character(len = *), intent(in), optional units,
            integer(i4), intent(in), optional missing_value,
            character(256), dimension(:, :), intent(in), optional attributes,
            logical, intent(in), optional create,
            integer(i4), intent(inout), optional ncid,
            integer(i4), intent(in), optional nrec )  [private]
```

References check(), close_netcdf(), ndims, and open_netcdf().

Here is the call graph for this function:

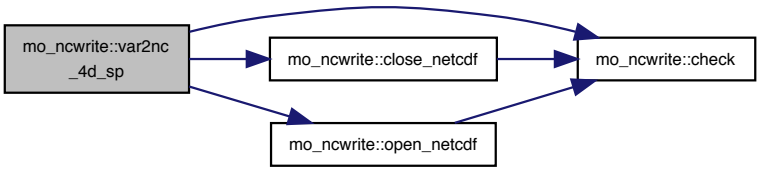


16.66.1.31 var2nc_4d_sp()

```
subroutine mo_ncwrite::var2nc_4d_sp (
             character(len = *), intent(in) f_name,
             real(sp), dimension(:, :, :, :), intent(in) arr,
             character(len = *), dimension(:), intent(in) dnames,
             character(len = *), intent(in) v_name,
             integer(i4), intent(in), optional dim_unlimited,
             character(len = *), intent(in), optional long_name,
             character(len = *), intent(in), optional units,
             real(sp), intent(in), optional missing_value,
             character(256), dimension(:, :), intent(in), optional attributes,
             logical, intent(in), optional create,
             integer(i4), intent(inout), optional ncid,
             integer(i4), intent(in), optional nrec )  [private]
```

References check(), close_netcdf(), ndims, and open_netcdf().

Here is the call graph for this function:



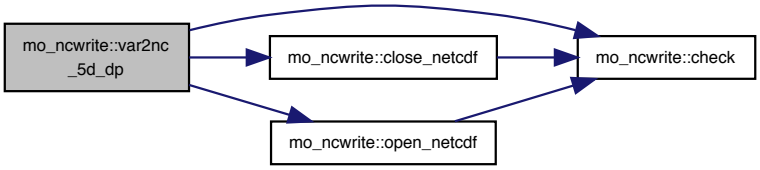
16.66.1.32 var2nc_5d_dp()

```
subroutine mo_ncwrite::var2nc_5d_dp (
             character(len = *), intent(in) f_name,
             real(dp), dimension(:, :, :, :, :), intent(in) arr,
             character(len = *), dimension(:), intent(in) dnames,
```

```
            character(len = *), intent(in) v_name,
            integer(i4), intent(in), optional dim_unlimited,
            character(len = *), intent(in), optional long_name,
            character(len = *), intent(in), optional units,
            real(dp), intent(in), optional missing_value,
            character(256), dimension(:, :), intent(in), optional attributes,
            logical, intent(in), optional create,
            integer(i4), intent(inout), optional ncid,
            integer(i4), intent(in), optional nrec )  [private]
```

References check(), close_netcdf(), ndims, and open_netcdf().

Here is the call graph for this function:

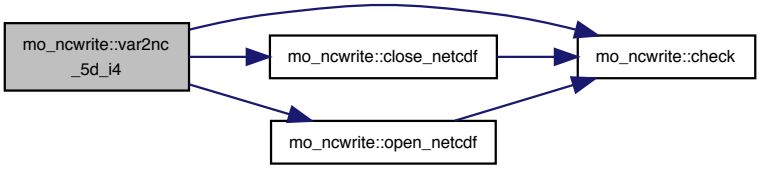


16.66.1.33 var2nc_5d_i4()

```
subroutine mo_ncwrite::var2nc_5d_i4 (
            character(len = *), intent(in) f_name,
            integer(i4), dimension(:, :, :, :, :), intent(in) arr,
            character(len = *), dimension(:), intent(in) dnames,
            character(len = *), intent(in) v_name,
            integer(i4), intent(in), optional dim_unlimited,
            character(len = *), intent(in), optional long_name,
            character(len = *), intent(in), optional units,
            integer(i4), intent(in), optional missing_value,
            character(256), dimension(:, :), intent(in), optional attributes,
            logical, intent(in), optional create,
            integer(i4), intent(inout), optional ncid,
            integer(i4), intent(in), optional nrec )  [private]
```

References check(), close_netcdf(), ndims, and open_netcdf().

Here is the call graph for this function:

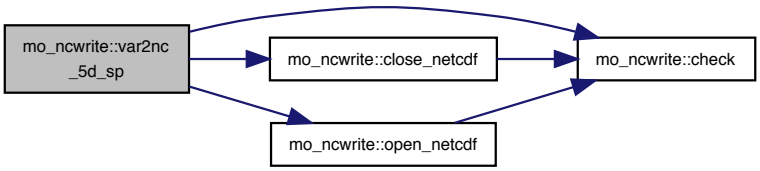


16.66.1.34 var2nc_5d_sp()

```
subroutine mo_ncwrite::var2nc_5d_sp (
             character(len = *), intent(in) f_name,
             real(sp), dimension(:, :, :, :, :), intent(in) arr,
             character(len = *), dimension(:), intent(in) dnames,
             character(len = *), intent(in) v_name,
             integer(i4), intent(in), optional dim_unlimited,
             character(len = *), intent(in), optional long_name,
             character(len = *), intent(in), optional units,
             real(sp), intent(in), optional missing_value,
             character(256), dimension(:, :), intent(in), optional attributes,
             logical, intent(in), optional create,
             integer(i4), intent(inout), optional ncid,
             integer(i4), intent(in), optional nrec )  [private]
```

References check(), close_netcdf(), ndims, and open_netcdf().

Here is the call graph for this function:

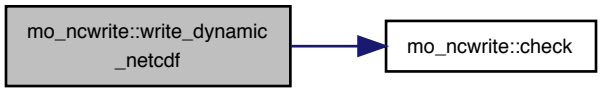


16.66.1.35 write_dynamic_netcdf()

```
subroutine, public mo_ncwrite::write_dynamic_netcdf (
             integer(i4), intent(in) ncId,
             integer(i4), intent(in) irec )
```

References check(), nvars, and v.

Here is the call graph for this function:

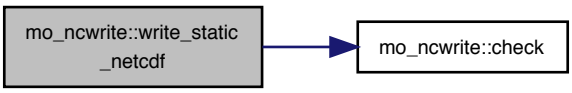


16.66.1.36 write_static_netcdf()

```
subroutine, public mo_ncwrite::write_static_netcdf (
             integer(i4), intent(in) ncId )
```

References check(), nvars, and v.

Here is the call graph for this function:



16.66.2 Variable Documentation

16.66.2.1 dnc

```
type (dims), dimension(:), allocatable, public mo_ncwrite::dnc
```

Referenced by create_netcdf(), and mo_set_netcdf_outputs::set_netcdf().

16.66.2.2 gatt

```
type(attribute), dimension(ngatt), public mo_ncwrite::gatt
```

Referenced by create_netcdf().

16.66.2.3 maxlen

```
integer(i4), parameter, public mo_ncwrite::maxlen = 256
```

16.66.2.4 nattdim

```
integer(i4), parameter, public mo_ncwrite::nattdim = 2
```

16.66.2.5 ndims

```
integer(i4), public mo_ncwrite::ndims
```

Referenced by create_netcdf(), dump_netcdf_1d_dp(), dump_netcdf_1d_i4(), dump_netcdf_1d_sp(), dump_netcdf_2d_dp(), dump_netcdf_2d_i4(), dump_netcdf_2d_sp(), dump_netcdf_3d_dp(), dump_netcdf_3d_i4(), dump_netcdf_3d_sp(), dump_netcdf_4d_dp(), dump_netcdf_4d_i4(), dump_netcdf_4d_sp(), dump_netcdf_5d_dp(), dump_netcdf_5d_i4(), dump_netcdf_5d_sp(), mo_set_netcdf_outputs::set_netcdf(), var2nc_1d_dp(), var2nc_1d_i4(), var2nc_1d_sp(), var2nc_2d_dp(), var2nc_2d_i4(), var2nc_2d_sp(), var2nc_3d_dp(), var2nc_3d_i4(), var2nc_3d_sp(), var2nc_4d_dp(), var2nc_4d_i4(), var2nc_4d_sp(), var2nc_5d_dp(), var2nc_5d_i4(), and var2nc_5d_sp().

16.66.2.6 ngatt

```
integer(i4), parameter, public mo_ncwrite::ngatt = 20
```

Referenced by create_netcdf().

16.66.2.7 nmaxatt

```
integer(i4), parameter, public mo_ncwrite::nmaxatt = 20
```

16.66.2.8 nmaxdim

```
integer(i4), parameter, public mo_ncwrite::nmaxdim = 5
```

16.66.2.9 nvars

```
integer(i4), public mo_ncwrite::nvars
```

Referenced by create_netcdf(), mo_set_netcdf_outputs::set_netcdf(), write_dynamic_netcdf(), and write_static_netcdf().

16.66.2.10 v

```
type(variable), dimension(:), allocatable, public mo_ncwrite::v
```

Referenced by create_netcdf(), mo_set_netcdf_outputs::set_netcdf(), write_dynamic_netcdf(), and write_static_↩ netcdf().

16.67 mo_netcdf Module Reference

NetCDF Fortran 90 interface wrapper.

Data Types

- interface ncdataset

  Provides basic file modification functionality.
- type ncdimension

  Provides the dimension access functionality.
- interface ncvariable

Functions/Subroutines

- subroutine initncvariable (self, id, parent)
- subroutine initncdimension (self, id, parent)
- subroutine initncdataset (self, fname, mode)
- type(ncvariable) function newncvariable (id, parent)
- type(ncdimension) function newncdimension (id, parent)
- type(ncdataset) function newncdataset (fname, mode)
- subroutine close (self)
- integer(i4) function getnovariables (self)
- integer(i4) function, dimension(:), allocatable getvariableids (self)
- type(ncvariable) function, dimension(:), allocatable getvariables (self)
- character(len=256) function getdimensionname (self)
- integer(i4) function getdimensionlength (self)
- logical function isdatasetunlimited (self)
- type(ncdimension) function getunlimiteddimension (self)
- logical function equalncdimensions (dim1, dim2)
- logical function isunlimiteddimension (self)
- type(ncdimension) function setdimension (self, name, length)
- logical function hasvariable (self, name)
- logical function hasdimension (self, name)
- type(ncvariable) function setvariablewithids (self, name, dtype, dimensions, contiguous, chunksizes, deflate↩ _level, shuffle, fletcher32, endianness, cache_size, cache_nelems, cache_preemption)
- type(ncvariable) function setvariablewithnames (self, name, dtype, dimensions, contiguous, chunksizes, deflate_level, shuffle, fletcher32, endianness, cache_size, cache_nelems, cache_preemption)
- type(ncvariable) function setvariablewithtypes (self, name, dtype, dimensions, contiguous, chunksizes, deflate_level, shuffle, fletcher32, endianness, cache_size, cache_nelems, cache_preemption)
- type(ncdimension) function getdimensionbyid (self, id)
- type(ncdimension) function getdimensionbyname (self, name)
- type(ncvariable) function getvariablebyname (self, name)
- character(len=256) function getvariablename (self)
- integer(i4) function getnodimensions (self)
- type(ncdimension) function, dimension(:), allocatable getvariabledimensions (self)
- integer(i4) function, dimension(:), allocatable getvariableshape (self)
- character(3) function getvariabledtype (self)
- logical function isunlimitedvariable (self)
- logical function hasattribute (self, name)

- subroutine setglobalattributechar (self, name, data)
- subroutine setglobalattributei8 (self, name, data)
- subroutine setglobalattributei16 (self, name, data)
- subroutine setglobalattributei32 (self, name, data)
- subroutine setglobalattributei64 (self, name, data)
- subroutine setglobalattributef32 (self, name, data)
- subroutine setglobalattributef64 (self, name, data)
- subroutine getglobalattributechar (self, name, avalue)
- subroutine getglobalattributei8 (self, name, avalue)
- subroutine getglobalattributei16 (self, name, avalue)
- subroutine getglobalattributei32 (self, name, avalue)
- subroutine getglobalattributei64 (self, name, avalue)
- subroutine getglobalattributef32 (self, name, avalue)
- subroutine getglobalattributef64 (self, name, avalue)
- subroutine setvariableattributechar (self, name, data)
- subroutine setvariableattributei8 (self, name, data)
- subroutine setvariableattributei16 (self, name, data)
- subroutine setvariableattributei32 (self, name, data)
- subroutine setvariableattributei64 (self, name, data)
- subroutine setvariableattributef32 (self, name, data)
- subroutine setvariableattributef64 (self, name, data)
- subroutine getvariableattributechar (self, name, avalue)
- subroutine getvariableattributei8 (self, name, avalue)
- subroutine getvariableattributei16 (self, name, avalue)
- subroutine getvariableattributei32 (self, name, avalue)
- subroutine getvariableattributei64 (self, name, avalue)
- subroutine getvariableattributef32 (self, name, avalue)
- subroutine getvariableattributef64 (self, name, avalue)
- subroutine setvariablefillvaluei8 (self, fvalue)
- subroutine setvariablefillvaluei16 (self, fvalue)
- subroutine setvariablefillvaluei32 (self, fvalue)
- subroutine setvariablefillvaluei64 (self, fvalue)
- subroutine setvariablefillvaluef32 (self, fvalue)
- subroutine setvariablefillvaluef64 (self, fvalue)
- subroutine getvariablefillvaluei8 (self, fvalue)
- subroutine getvariablefillvaluei16 (self, fvalue)
- subroutine getvariablefillvaluei32 (self, fvalue)
- subroutine getvariablefillvaluei64 (self, fvalue)
- subroutine getvariablefillvaluef32 (self, fvalue)
- subroutine getvariablefillvaluef64 (self, fvalue)
- subroutine setdatascalari8 (self, values, start)
- subroutine setdata1di8 (self, values, start, cnt, stride, map)
- subroutine setdata2di8 (self, values, start, cnt, stride, map)
- subroutine setdata3di8 (self, values, start, cnt, stride, map)
- subroutine setdata4di8 (self, values, start, cnt, stride, map)
- subroutine setdata5di8 (self, values, start, cnt, stride, map)
- subroutine setdatascalari16 (self, values, start)
- subroutine setdata1di16 (self, values, start, cnt, stride, map)
- subroutine setdata2di16 (self, values, start, cnt, stride, map)
- subroutine setdata3di16 (self, values, start, cnt, stride, map)
- subroutine setdata4di16 (self, values, start, cnt, stride, map)
- subroutine setdata5di16 (self, values, start, cnt, stride, map)
- subroutine setdatascalari32 (self, values, start)
- subroutine setdata1di32 (self, values, start, cnt, stride, map)
- subroutine setdata2di32 (self, values, start, cnt, stride, map)

- subroutine setdata3di32 (self, values, start, cnt, stride, map)
- subroutine setdata4di32 (self, values, start, cnt, stride, map)
- subroutine setdata5di32 (self, values, start, cnt, stride, map)
- subroutine setdatascalari64 (self, values, start)
- subroutine setdata1di64 (self, values, start, cnt, stride, map)
- subroutine setdata2di64 (self, values, start, cnt, stride, map)
- subroutine setdata3di64 (self, values, start, cnt, stride, map)
- subroutine setdata4di64 (self, values, start, cnt, stride, map)
- subroutine setdata5di64 (self, values, start, cnt, stride, map)
- subroutine setdatascalarf32 (self, values, start)
- subroutine setdata1df32 (self, values, start, cnt, stride, map)
- subroutine setdata2df32 (self, values, start, cnt, stride, map)
- subroutine setdata3df32 (self, values, start, cnt, stride, map)
- subroutine setdata4df32 (self, values, start, cnt, stride, map)
- subroutine setdata5df32 (self, values, start, cnt, stride, map)
- subroutine setdatascalarf64 (self, values, start)
- subroutine setdata1df64 (self, values, start, cnt, stride, map)
- subroutine setdata2df64 (self, values, start, cnt, stride, map)
- subroutine setdata3df64 (self, values, start, cnt, stride, map)
- subroutine setdata4df64 (self, values, start, cnt, stride, map)
- subroutine setdata5df64 (self, values, start, cnt, stride, map)
- subroutine getdatascalari8 (self, data, start, cnt, stride, map)
- subroutine getdata1di8 (self, data, start, cnt, stride, map)
- subroutine getdata2di8 (self, data, start, cnt, stride, map)
- subroutine getdata3di8 (self, data, start, cnt, stride, map)
- subroutine getdata4di8 (self, data, start, cnt, stride, map)
- subroutine getdata5di8 (self, data, start, cnt, stride, map)
- subroutine getdatascalari16 (self, data, start, cnt, stride, map)
- subroutine getdata1di16 (self, data, start, cnt, stride, map)
- subroutine getdata2di16 (self, data, start, cnt, stride, map)
- subroutine getdata3di16 (self, data, start, cnt, stride, map)
- subroutine getdata4di16 (self, data, start, cnt, stride, map)
- subroutine getdata5di16 (self, data, start, cnt, stride, map)
- subroutine getdatascalari32 (self, data, start, cnt, stride, map)
- subroutine getdata1di32 (self, data, start, cnt, stride, map)
- subroutine getdata2di32 (self, data, start, cnt, stride, map)
- subroutine getdata3di32 (self, data, start, cnt, stride, map)
- subroutine getdata4di32 (self, data, start, cnt, stride, map)
- subroutine getdata5di32 (self, data, start, cnt, stride, map)
- subroutine getdatascalari64 (self, data, start, cnt, stride, map)
- subroutine getdata1di64 (self, data, start, cnt, stride, map)
- subroutine getdata2di64 (self, data, start, cnt, stride, map)
- subroutine getdata3di64 (self, data, start, cnt, stride, map)
- subroutine getdata4di64 (self, data, start, cnt, stride, map)
- subroutine getdata5di64 (self, data, start, cnt, stride, map)
- subroutine getdatascalarf32 (self, data, start, cnt, stride, map)
- subroutine getdata1df32 (self, data, start, cnt, stride, map)
- subroutine getdata2df32 (self, data, start, cnt, stride, map)
- subroutine getdata3df32 (self, data, start, cnt, stride, map)
- subroutine getdata4df32 (self, data, start, cnt, stride, map)
- subroutine getdata5df32 (self, data, start, cnt, stride, map)
- subroutine getdatascalarf64 (self, data, start, cnt, stride, map)
- subroutine getdata1df64 (self, data, start, cnt, stride, map)
- subroutine getdata2df64 (self, data, start, cnt, stride, map)
- subroutine getdata3df64 (self, data, start, cnt, stride, map)

- subroutine getdata4df64 (self, data, start, cnt, stride, map)
- subroutine getdata5df64 (self, data, start, cnt, stride, map)
- integer(i4) function, dimension(datarank) getreaddatashape (var, datarank, instart, incnt, instride)
- integer(i4) function getdtypefromstring (dtype)
- character(3) function getdtypefrominteger (dtype)
- subroutine check (status, msg)

16.67.1 Detailed Description

NetCDF Fortran 90 interface wrapper.

A thin wrapper around the NetCDF Fortran 90 interface. Provided are currently 3 user facing derived Types:

1. NcDataset
2. NcDimension
3. NcVariable

Authors

David Schaefer

Date

Jun 2015

16.67.2 Function/Subroutine Documentation

16.67.2.1 check()

```
subroutine mo_netcdf::check (
             integer(i4), intent(in) status,
             character(*), intent(in) msg )
```

Referenced by close(), getdata1df32(), getdata1df64(), getdata1di16(), getdata1di32(), getdata1di64(), getdata1di8(), getdata2df32(), getdata2df64(), getdata2di16(), getdata2di32(), getdata2di64(), getdata2di8(), getdata3df32(), getdata3df64(), getdata3di16(), getdata3di32(), getdata3di64(), getdata3di8(), getdata4df32(), getdata4df64(), getdata4di16(), getdata4di32(), getdata4di64(), getdata4di8(), getdata5df32(), getdata5df64(), getdata5di16(), getdata5di32(), getdata5di64(), getdata5di8(), getdatascalarf32(), getdatascalarf64(), getdatascalari16(), getdatascalari32(), getdatascalari64(), getdatascalari8(), getdimensionbyid(), getdimensionbyname(), getdimensionlength(), getdimensionname(), getglobalattributechar(), getglobalattributef32(), getglobalattributef64(), getglobalattributei16(), getglobalattributei32(), getglobalattributei64(), getglobalattributei8(), getnodimensions(), getnovariables(), getunlimiteddimension(), getvariableattributechar(), getvariableattributef32(), getvariableattributef64(), getvariableattributei16(), getvariableattributei32(), getvariableattributei64(), getvariableattributei8(), getvariablebyname(), getvariabledimensions(), getvariabledtype(), getvariableids(), getvariablename(), initncdataset(), isdatasetunlimited(), setdata1df32(), setdata1df64(), setdata1di16(), setdata1di32(), setdata1di64(), setdata1di8(), setdata2df32(), setdata2df64(), setdata2di16(), setdata2di32(), setdata2di64(), setdata2di8(), setdata3df32(), setdata3df64(), setdata3di16(), setdata3di32(), setdata3di64(), setdata3di8(), setdata4df32(), setdata4df64(), setdata4di16(), setdata4di32(), setdata4di64(), setdata4di8(), setdata5df32(), setdata5df64(), setdata5di16(), setdata5di32(), setdata5di64(), setdata5di8(), setdatascalarf32(), setdatascalarf64(), setdatascalari16(), setdatascalari32(), setdatascalari64(), setdatascalari8(), setdimension(), setglobalattributechar(), setglobalattributef32(), setglobalattributef64(), setglobalattributei16(), setglobalattributei32(), setglobalattributei64(), setglobalattributei8(), setvariableattributechar(), setvariableattributef32(), setvariableattributef64(), setvariableattributei16(), setvariableattributei32(), setvariableattributei64(), setvariableattributei8(), and setvariablewithids().

16.67.2.2 close()

```
subroutine mo_netcdf::close (
             class(ncdataset) self )
```

References check().

Here is the call graph for this function:

16.67.2.3 equalncdimensions()

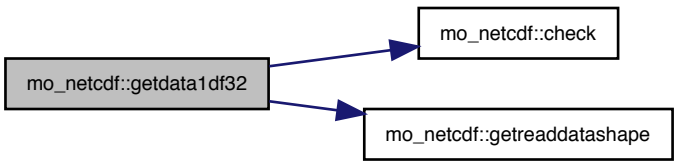
```
logical function mo_netcdf::equalncdimensions (
             type(ncdimension), intent(in) dim1,
             type(ncdimension), intent(in) dim2 )
```

16.67.2.4 getdata1df32()

```
subroutine mo_netcdf::getdata1df32 (
             class(ncvariable), intent(in) self,
             real(sp), dimension(:), intent(out), allocatable data,
             integer(i4), dimension(:), intent(in), optional start,
             integer(i4), dimension(:), intent(in), optional cnt,
             integer(i4), dimension(:), intent(in), optional stride,
             integer(i4), dimension(:), intent(in), optional map )
```

References check(), and getreaddatashape().

Here is the call graph for this function:

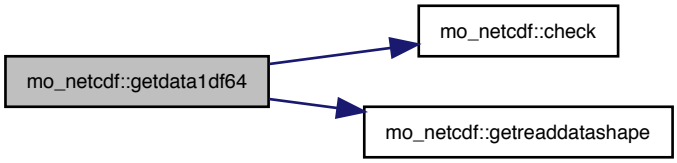


16.67.2.5 getdata1df64()

```
subroutine mo_netcdf::getdata1df64 (
             class(ncvariable), intent(in) self,
             real(dp), dimension(:), intent(out), allocatable data,
             integer(i4), dimension(:), intent(in), optional start,
             integer(i4), dimension(:), intent(in), optional cnt,
             integer(i4), dimension(:), intent(in), optional stride,
             integer(i4), dimension(:), intent(in), optional map )
```

References check(), and getreaddatashape().

Here is the call graph for this function:

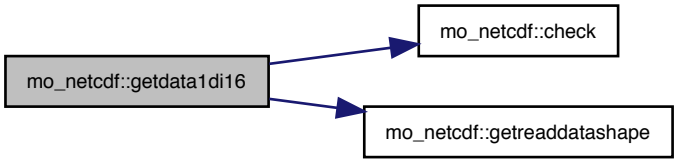


16.67.2.6 getdata1di16()

```
subroutine mo_netcdf::getdata1di16 (
             class(ncvariable), intent(in) self,
             integer(i2), dimension(:), intent(out), allocatable data,
             integer(i4), dimension(:), intent(in), optional start,
             integer(i4), dimension(:), intent(in), optional cnt,
             integer(i4), dimension(:), intent(in), optional stride,
             integer(i4), dimension(:), intent(in), optional map )
```

References check(), and getreaddatashape().

Here is the call graph for this function:

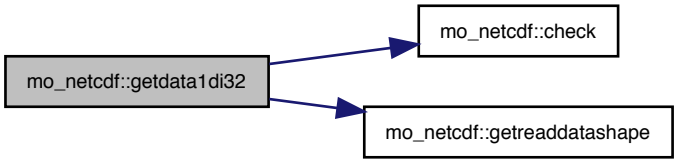


16.67.2.7 getdata1di32()

```
subroutine mo_netcdf::getdata1di32 (
             class(ncvariable), intent(in) self,
             integer(i4), dimension(:), intent(out), allocatable data,
             integer(i4), dimension(:), intent(in), optional start,
             integer(i4), dimension(:), intent(in), optional cnt,
             integer(i4), dimension(:), intent(in), optional stride,
             integer(i4), dimension(:), intent(in), optional map )
```

References check(), and getreaddatashape().

Here is the call graph for this function:

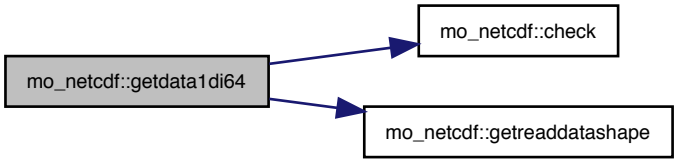


16.67.2.8 getdata1di64()

```
subroutine mo_netcdf::getdata1di64 (
             class(ncvariable), intent(in) self,
             integer(i8), dimension(:), intent(out), allocatable data,
             integer(i4), dimension(:), intent(in), optional start,
             integer(i4), dimension(:), intent(in), optional cnt,
             integer(i4), dimension(:), intent(in), optional stride,
             integer(i4), dimension(:), intent(in), optional map )
```

References check(), and getreaddatashape().

Here is the call graph for this function:

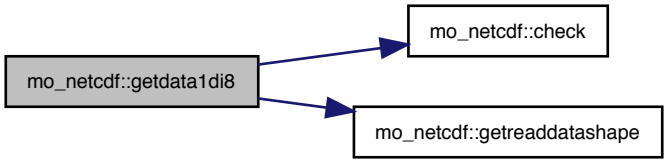


16.67.2.9 getdata1di8()

```
subroutine mo_netcdf::getdata1di8 (
             class(ncvariable), intent(in) self,
             integer(i1), dimension(:), intent(out), allocatable data,
             integer(i4), dimension(:), intent(in), optional start,
             integer(i4), dimension(:), intent(in), optional cnt,
             integer(i4), dimension(:), intent(in), optional stride,
             integer(i4), dimension(:), intent(in), optional map )
```

References check(), and getreaddatashape().

Here is the call graph for this function:

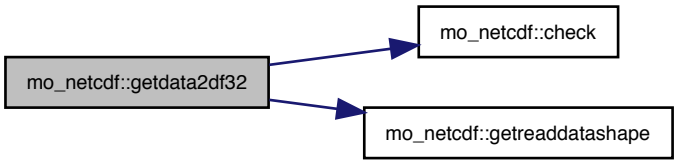


16.67.2.10 getdata2df32()

```
subroutine mo_netcdf::getdata2df32 (
             class(ncvariable), intent(in) self,
             real(sp), dimension(:, :), intent(out), allocatable data,
             integer(i4), dimension(:), intent(in), optional start,
             integer(i4), dimension(:), intent(in), optional cnt,
             integer(i4), dimension(:), intent(in), optional stride,
             integer(i4), dimension(:), intent(in), optional map )
```

References check(), and getreaddatashape().

Here is the call graph for this function:

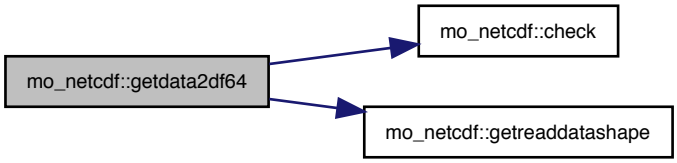


16.67.2.11 getdata2df64()

```
subroutine mo_netcdf::getdata2df64 (
            class(ncvariable), intent(in) self,
            real(dp), dimension(:, :), intent(out), allocatable data,
            integer(i4), dimension(:), intent(in), optional start,
            integer(i4), dimension(:), intent(in), optional cnt,
            integer(i4), dimension(:), intent(in), optional stride,
            integer(i4), dimension(:), intent(in), optional map )
```

References check(), and getreaddatashape().

Here is the call graph for this function:

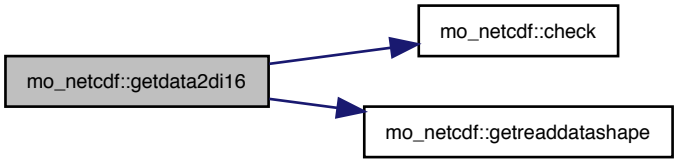


16.67.2.12 getdata2di16()

```
subroutine mo_netcdf::getdata2di16 (
            class(ncvariable), intent(in) self,
            integer(i2), dimension(:, :), intent(out), allocatable data,
            integer(i4), dimension(:), intent(in), optional start,
            integer(i4), dimension(:), intent(in), optional cnt,
            integer(i4), dimension(:), intent(in), optional stride,
            integer(i4), dimension(:), intent(in), optional map )
```

References check(), and getreaddatashape().

Here is the call graph for this function:

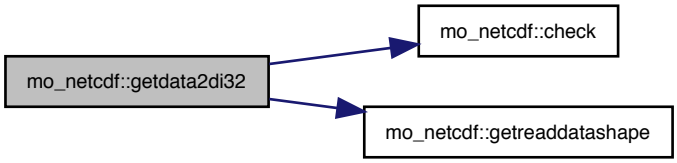


16.67.2.13 getdata2di32()

```
subroutine mo_netcdf::getdata2di32 (
             class(ncvariable), intent(in) self,
             integer(i4), dimension(:, :), intent(out), allocatable data,
             integer(i4), dimension(:), intent(in), optional start,
             integer(i4), dimension(:), intent(in), optional cnt,
             integer(i4), dimension(:), intent(in), optional stride,
             integer(i4), dimension(:), intent(in), optional map )
```

References check(), and getreaddatashape().

Here is the call graph for this function:

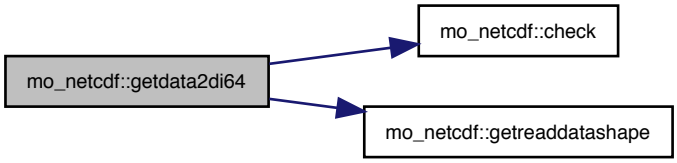


16.67.2.14 getdata2di64()

```
subroutine mo_netcdf::getdata2di64 (
             class(ncvariable), intent(in) self,
             integer(i8), dimension(:, :), intent(out), allocatable data,
             integer(i4), dimension(:), intent(in), optional start,
             integer(i4), dimension(:), intent(in), optional cnt,
             integer(i4), dimension(:), intent(in), optional stride,
             integer(i4), dimension(:), intent(in), optional map )
```

References check(), and getreaddatashape().

Here is the call graph for this function:

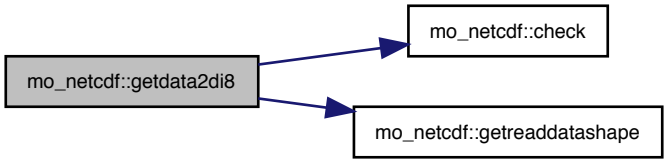


16.67.2.15 getdata2di8()

```
subroutine mo_netcdf::getdata2di8 (
             class(ncvariable), intent(in) self,
             integer(i1), dimension(:, :), intent(out), allocatable data,
             integer(i4), dimension(:), intent(in), optional start,
             integer(i4), dimension(:), intent(in), optional cnt,
             integer(i4), dimension(:), intent(in), optional stride,
             integer(i4), dimension(:), intent(in), optional map )
```

References check(), and getreaddatashape().

Here is the call graph for this function:

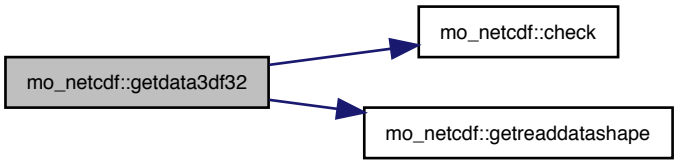


16.67.2.16 getdata3df32()

```
subroutine mo_netcdf::getdata3df32 (
             class(ncvariable), intent(in) self,
             real(sp), dimension(:, :, :), intent(out), allocatable data,
             integer(i4), dimension(:), intent(in), optional start,
             integer(i4), dimension(:), intent(in), optional cnt,
             integer(i4), dimension(:), intent(in), optional stride,
             integer(i4), dimension(:), intent(in), optional map )
```

References check(), and getreaddatashape().

Here is the call graph for this function:

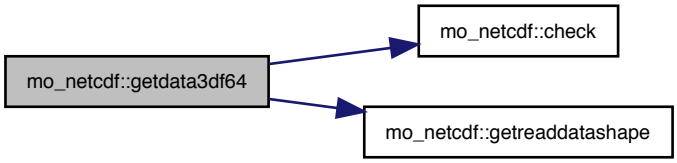


16.67.2.17 getdata3df64()

```
subroutine mo_netcdf::getdata3df64 (
             class(ncvariable), intent(in) self,
             real(dp), dimension(:, :, :), intent(out), allocatable data,
             integer(i4), dimension(:), intent(in), optional start,
             integer(i4), dimension(:), intent(in), optional cnt,
             integer(i4), dimension(:), intent(in), optional stride,
             integer(i4), dimension(:), intent(in), optional map )
```

References check(), and getreaddatashape().

Here is the call graph for this function:

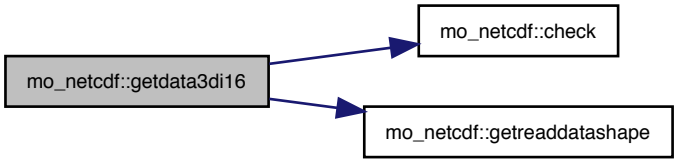


16.67.2.18 getdata3di16()

```
subroutine mo_netcdf::getdata3di16 (
             class(ncvariable), intent(in) self,
             integer(i2), dimension(:, :, :), intent(out), allocatable data,
             integer(i4), dimension(:), intent(in), optional start,
             integer(i4), dimension(:), intent(in), optional cnt,
             integer(i4), dimension(:), intent(in), optional stride,
             integer(i4), dimension(:), intent(in), optional map )
```

References check(), and getreaddatashape().

Here is the call graph for this function:

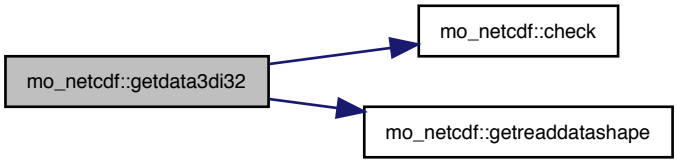


16.67.2.19 getdata3di32()

```
subroutine mo_netcdf::getdata3di32 (
             class(ncvariable), intent(in) self,
             integer(i4), dimension(:, :, :), intent(out), allocatable data,
             integer(i4), dimension(:), intent(in), optional start,
             integer(i4), dimension(:), intent(in), optional cnt,
             integer(i4), dimension(:), intent(in), optional stride,
             integer(i4), dimension(:), intent(in), optional map )
```

References check(), and getreaddatashape().

Here is the call graph for this function:

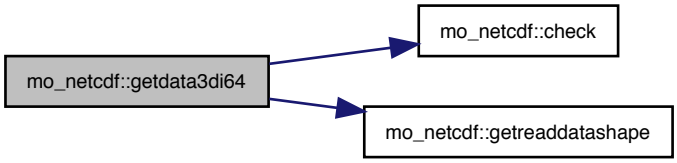


16.67.2.20 getdata3di64()

```
subroutine mo_netcdf::getdata3di64 (
             class(ncvariable), intent(in) self,
             integer(i8), dimension(:, :, :), intent(out), allocatable data,
             integer(i4), dimension(:), intent(in), optional start,
             integer(i4), dimension(:), intent(in), optional cnt,
             integer(i4), dimension(:), intent(in), optional stride,
             integer(i4), dimension(:), intent(in), optional map )
```

References check(), and getreaddatashape().

Here is the call graph for this function:

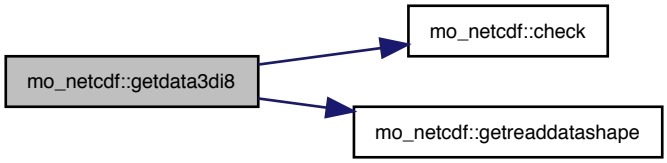


16.67.2.21 getdata3di8()

```
subroutine mo_netcdf::getdata3di8 (
             class(ncvariable), intent(in) self,
             integer(i1), dimension(:, :, :), intent(out), allocatable data,
             integer(i4), dimension(:), intent(in), optional start,
             integer(i4), dimension(:), intent(in), optional cnt,
             integer(i4), dimension(:), intent(in), optional stride,
             integer(i4), dimension(:), intent(in), optional map )
```

References check(), and getreaddatashape().

Here is the call graph for this function:

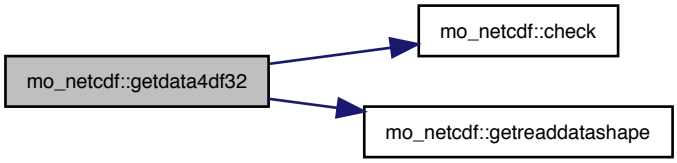


16.67.2.22 getdata4df32()

```
subroutine mo_netcdf::getdata4df32 (
             class(ncvariable), intent(in) self,
             real(sp), dimension(:, :, :, :), intent(out), allocatable data,
             integer(i4), dimension(:), intent(in), optional start,
             integer(i4), dimension(:), intent(in), optional cnt,
             integer(i4), dimension(:), intent(in), optional stride,
             integer(i4), dimension(:), intent(in), optional map )
```

References check(), and getreaddatashape().

Here is the call graph for this function:

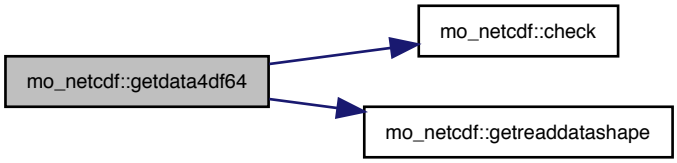


16.67.2.23 getdata4df64()

```
subroutine mo_netcdf::getdata4df64 (
             class(ncvariable), intent(in) self,
             real(dp), dimension(:, :, :, :), intent(out), allocatable data,
             integer(i4), dimension(:), intent(in), optional start,
             integer(i4), dimension(:), intent(in), optional cnt,
             integer(i4), dimension(:), intent(in), optional stride,
             integer(i4), dimension(:), intent(in), optional map )
```

References check(), and getreaddatashape().

Here is the call graph for this function:

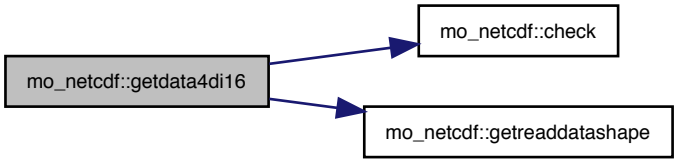


16.67.2.24 getdata4di16()

```
subroutine mo_netcdf::getdata4di16 (
             class(ncvariable), intent(in) self,
             integer(i2), dimension(:, :, :, :), intent(out), allocatable data,
             integer(i4), dimension(:), intent(in), optional start,
             integer(i4), dimension(:), intent(in), optional cnt,
             integer(i4), dimension(:), intent(in), optional stride,
             integer(i4), dimension(:), intent(in), optional map )
```

References check(), and getreaddatashape().

Here is the call graph for this function:

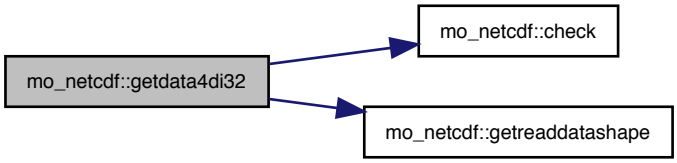


16.67.2.25 getdata4di32()

```
subroutine mo_netcdf::getdata4di32 (
             class(ncvariable), intent(in) self,
             integer(i4), dimension(:, :, :, :), intent(out), allocatable data,
             integer(i4), dimension(:), intent(in), optional start,
             integer(i4), dimension(:), intent(in), optional cnt,
             integer(i4), dimension(:), intent(in), optional stride,
             integer(i4), dimension(:), intent(in), optional map )
```

References check(), and getreaddatashape().

Here is the call graph for this function:

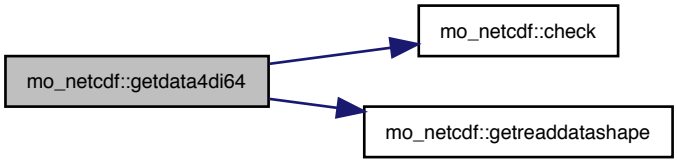


16.67.2.26 getdata4di64()

```
subroutine mo_netcdf::getdata4di64 (
             class(ncvariable), intent(in) self,
             integer(i8), dimension(:, :, :, :), intent(out), allocatable data,
             integer(i4), dimension(:), intent(in), optional start,
             integer(i4), dimension(:), intent(in), optional cnt,
             integer(i4), dimension(:), intent(in), optional stride,
             integer(i4), dimension(:), intent(in), optional map )
```

References check(), and getreaddatashape().

Here is the call graph for this function:

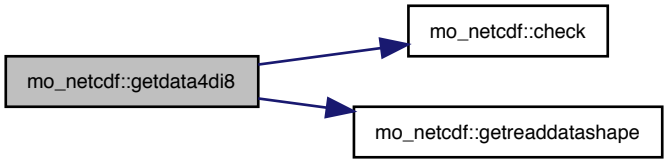


16.67.2.27 getdata4di8()

```
subroutine mo_netcdf::getdata4di8 (
             class(ncvariable), intent(in) self,
             integer(i1), dimension(:, :, :, :), intent(out), allocatable data,
             integer(i4), dimension(:), intent(in), optional start,
             integer(i4), dimension(:), intent(in), optional cnt,
             integer(i4), dimension(:), intent(in), optional stride,
             integer(i4), dimension(:), intent(in), optional map )
```

References check(), and getreaddatashape().

Here is the call graph for this function:

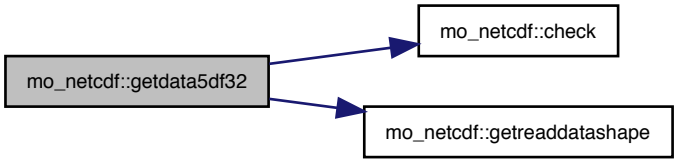


16.67.2.28 getdata5df32()

```
subroutine mo_netcdf::getdata5df32 (
             class(ncvariable), intent(in) self,
             real(sp), dimension(:, :, :, :, :), intent(out), allocatable data,
             integer(i4), dimension(:), intent(in), optional start,
             integer(i4), dimension(:), intent(in), optional cnt,
             integer(i4), dimension(:), intent(in), optional stride,
             integer(i4), dimension(:), intent(in), optional map )
```

References check(), and getreaddatashape().

Here is the call graph for this function:

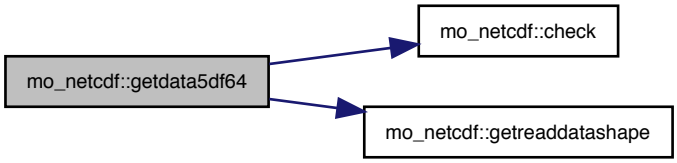


16.67.2.29 getdata5df64()

```
subroutine mo_netcdf::getdata5df64 (
            class(ncvariable), intent(in) self,
            real(dp), dimension(:, :, :, :, :), intent(out), allocatable data,
            integer(i4), dimension(:), intent(in), optional start,
            integer(i4), dimension(:), intent(in), optional cnt,
            integer(i4), dimension(:), intent(in), optional stride,
            integer(i4), dimension(:), intent(in), optional map )
```

References check(), and getreaddatashape().

Here is the call graph for this function:

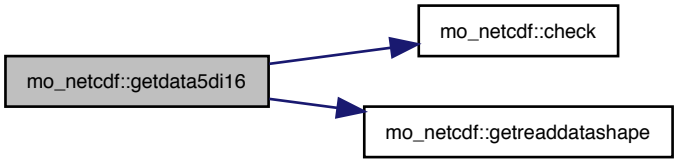


16.67.2.30 getdata5di16()

```
subroutine mo_netcdf::getdata5di16 (
            class(ncvariable), intent(in) self,
            integer(i2), dimension(:, :, :, :, :), intent(out), allocatable data,
            integer(i4), dimension(:), intent(in), optional start,
            integer(i4), dimension(:), intent(in), optional cnt,
            integer(i4), dimension(:), intent(in), optional stride,
            integer(i4), dimension(:), intent(in), optional map )
```

References check(), and getreaddatashape().

Here is the call graph for this function:

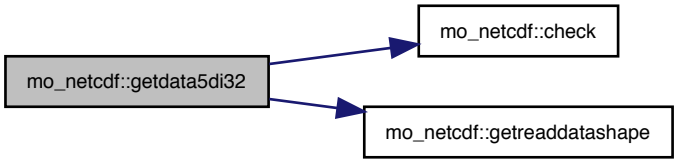


16.67.2.31 getdata5di32()

```
subroutine mo_netcdf::getdata5di32 (
            class(ncvariable), intent(in) self,
            integer(i4), dimension(:, :, :, :, :), intent(out), allocatable data,
            integer(i4), dimension(:), intent(in), optional start,
            integer(i4), dimension(:), intent(in), optional cnt,
            integer(i4), dimension(:), intent(in), optional stride,
            integer(i4), dimension(:), intent(in), optional map )
```

References check(), and getreaddatashape().

Here is the call graph for this function:

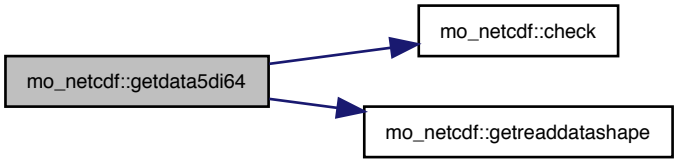


16.67.2.32 getdata5di64()

```
subroutine mo_netcdf::getdata5di64 (
            class(ncvariable), intent(in) self,
            integer(i8), dimension(:, :, :, :, :), intent(out), allocatable data,
            integer(i4), dimension(:), intent(in), optional start,
            integer(i4), dimension(:), intent(in), optional cnt,
            integer(i4), dimension(:), intent(in), optional stride,
            integer(i4), dimension(:), intent(in), optional map )
```

References check(), and getreaddatashape().

Here is the call graph for this function:

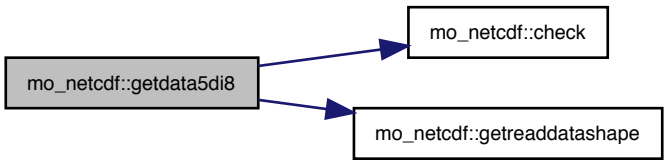


16.67.2.33 getdata5di8()

```
subroutine mo_netcdf::getdata5di8 (
            class(ncvariable), intent(in) self,
            integer(i1), dimension(:, :, :, :, :), intent(out), allocatable data,
            integer(i4), dimension(:), intent(in), optional start,
            integer(i4), dimension(:), intent(in), optional cnt,
            integer(i4), dimension(:), intent(in), optional stride,
            integer(i4), dimension(:), intent(in), optional map )
```

References check(), and getreaddatashape().

Here is the call graph for this function:

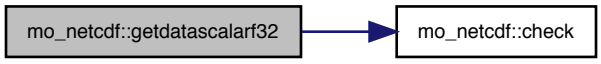


16.67.2.34 getdatascalarf32()

```
subroutine mo_netcdf::getdatascalarf32 (
            class(ncvariable), intent(in) self,
            real(sp), intent(out) data,
            integer(i4), dimension(:), intent(in), optional start,
            integer(i4), dimension(:), intent(in), optional cnt,
            integer(i4), dimension(:), intent(in), optional stride,
            integer(i4), dimension(:), intent(in), optional map )
```

References check().

Here is the call graph for this function:



16.67.2.35 getdatascalarf64()

```
subroutine mo_netcdf::getdatascalarf64 (
```

```
            class(ncvariable), intent(in) self,
            real(dp), intent(out) data,
            integer(i4), dimension(:), intent(in), optional start,
            integer(i4), dimension(:), intent(in), optional cnt,
            integer(i4), dimension(:), intent(in), optional stride,
            integer(i4), dimension(:), intent(in), optional map )
```

References check().

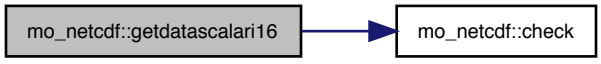
Here is the call graph for this function:

16.67.2.36 getdatascalari16()

```
subroutine mo_netcdf::getdatascalari16 (
            class(ncvariable), intent(in) self,
            integer(i2), intent(out) data,
            integer(i4), dimension(:), intent(in), optional start,
            integer(i4), dimension(:), intent(in), optional cnt,
            integer(i4), dimension(:), intent(in), optional stride,
            integer(i4), dimension(:), intent(in), optional map )
```

References check().

Here is the call graph for this function:

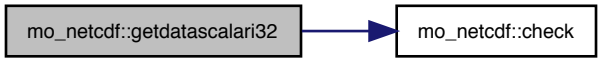


16.67.2.37 getdatascalari32()

```
subroutine mo_netcdf::getdatascalari32 (
            class(ncvariable), intent(in) self,
            integer(i4), intent(out) data,
            integer(i4), dimension(:), intent(in), optional start,
            integer(i4), dimension(:), intent(in), optional cnt,
            integer(i4), dimension(:), intent(in), optional stride,
            integer(i4), dimension(:), intent(in), optional map )
```

References check().

Here is the call graph for this function:

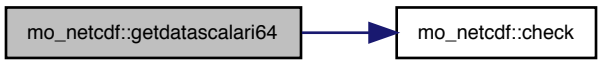


16.67.2.38 getdatascalari64()

```
subroutine mo_netcdf::getdatascalari64 (
             class(ncvariable), intent(in) self,
             integer(i8), intent(out) data,
             integer(i4), dimension(:), intent(in), optional start,
             integer(i4), dimension(:), intent(in), optional cnt,
             integer(i4), dimension(:), intent(in), optional stride,
             integer(i4), dimension(:), intent(in), optional map )
```

References check().

Here is the call graph for this function:

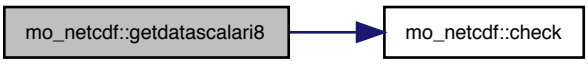


16.67.2.39 getdatascalari8()

```
subroutine mo_netcdf::getdatascalari8 (
             class(ncvariable), intent(in) self,
             integer(i1), intent(out) data,
             integer(i4), dimension(:), intent(in), optional start,
             integer(i4), dimension(:), intent(in), optional cnt,
             integer(i4), dimension(:), intent(in), optional stride,
             integer(i4), dimension(:), intent(in), optional map )
```

References check().

Here is the call graph for this function:

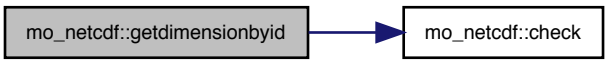


16.67.2.40 getdimensionbyid()

```
type(ncdimension) function mo_netcdf::getdimensionbyid (
            class(ncdataset), intent(in) self,
            integer(i4) id )
```

References check().

Here is the call graph for this function:

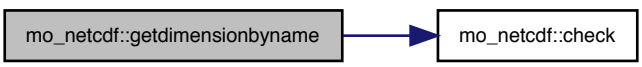


16.67.2.41 getdimensionbyname()

```
type(ncdimension) function mo_netcdf::getdimensionbyname (
            class(ncdataset), intent(in) self,
            character(*) name )
```

References check().

Here is the call graph for this function:

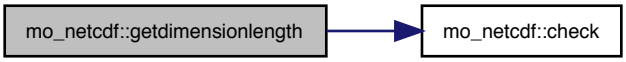


16.67.2.42 getdimensionlength()

```
integer(i4) function mo_netcdf::getdimensionlength (
            class(ncdimension), intent(in) self )
```

References check().

Here is the call graph for this function:

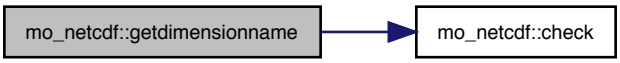


16.67.2.43 getdimensionname()

```
character(len = 256) function mo_netcdf::getdimensionname (
            class(ncdimension), intent(in) self )
```

References check().

Here is the call graph for this function:



16.67.2.44 getdtypefrominteger()

```
character(3) function mo_netcdf::getdtypefrominteger (
            integer(i4) dtype )
```

Referenced by getvariabledtype().

Here is the caller graph for this function:

16.67.2.45 getdtypefromstring()

```
integer(i4) function mo_netcdf::getdtypefromstring (
             character(*) dtype )
```

Referenced by setvariablewithids().

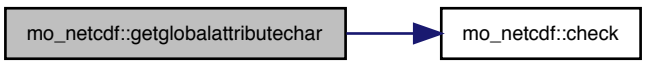
Here is the caller graph for this function:

16.67.2.46 getglobalattributechar()

```
subroutine mo_netcdf::getglobalattributechar (
             class(ncdataset), intent(in) self,
             character(*), intent(in) name,
             character(*), intent(out) avalue )
```

References check().

Here is the call graph for this function:

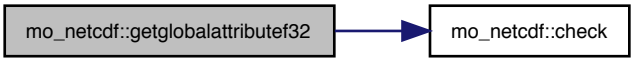


16.67.2.47 getglobalattributef32()

```
subroutine mo_netcdf::getglobalattributef32 (
             class(ncdataset), intent(in) self,
             character(*), intent(in) name,
             real(sp), intent(out) avalue )
```

References check().

Here is the call graph for this function:

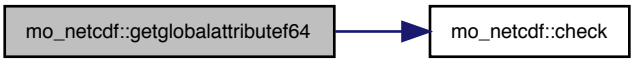


16.67.2.48 getglobalattributef64()

```
subroutine mo_netcdf::getglobalattributef64 (
             class(ncdataset), intent(in) self,
             character(*), intent(in) name,
             real(dp), intent(out) avalue )
```

References check().

Here is the call graph for this function:



16.67.2.49 getglobalattributei16()

```
subroutine mo_netcdf::getglobalattributei16 (
             class(ncdataset), intent(in) self,
             character(*), intent(in) name,
             integer(i2), intent(out) avalue )
```

References check().

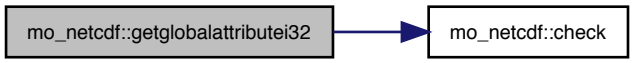
Here is the call graph for this function:

16.67.2.50 getglobalattributei32()

```
subroutine mo_netcdf::getglobalattributei32 (
             class(ncdataset), intent(in) self,
             character(*), intent(in) name,
             integer(i4), intent(out) avalue )
```

References check().

Here is the call graph for this function:



16.67.2.51 getglobalattributei64()

```
subroutine mo_netcdf::getglobalattributei64 (
             class(ncdataset), intent(in) self,
             character(*), intent(in) name,
             integer(i8), intent(out) avalue )
```

References check().

Here is the call graph for this function:

16.67.2.52 getglobalattributei8()

```
subroutine mo_netcdf::getglobalattributei8 (
             class(ncdataset), intent(in) self,
             character(*), intent(in) name,
             integer(i1), intent(out) avalue )
```

References check().

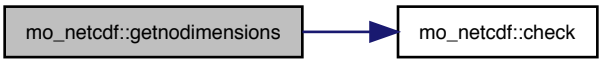
Here is the call graph for this function:

16.67.2.53 getnodimensions()

```
integer(i4) function mo_netcdf::getnodimensions (
             class(ncvariable), intent(in) self )
```

References check().

Here is the call graph for this function:

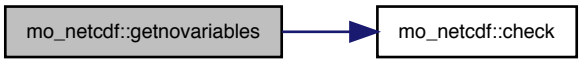


16.67.2.54 getnovariables()

```
integer(i4) function mo_netcdf::getnovariables (
             class(ncdataset), intent(in) self )
```

References check().

Here is the call graph for this function:

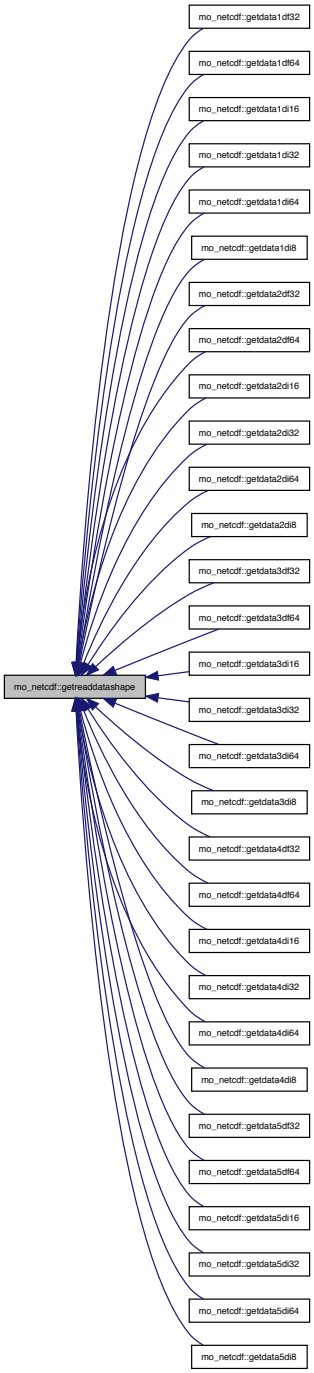


16.67.2.55 getreaddatashape()

```
integer(i4) function, dimension(datarank) mo_netcdf::getreaddatashape (
             type(ncvariable), intent(in) var,
             integer(i4), intent(in) datarank,
             integer(i4), dimension(:), intent(in), optional instart,
             integer(i4), dimension(:), intent(in), optional incnt,
             integer(i4), dimension(:), intent(in), optional instride )
```

Referenced by getdata1df32(), getdata1df64(), getdata1di16(), getdata1di32(), getdata1di64(), getdata1di8(), getdata2df32(), getdata2df64(), getdata2di16(), getdata2di32(), getdata2di64(), getdata2di8(), getdata3df32(), getdata3df64(), getdata3di16(), getdata3di32(), getdata3di64(), getdata3di8(), getdata4df32(), getdata4df64(), getdata4di16(), getdata4di32(), getdata4di64(), getdata4di8(), getdata5df32(), getdata5df64(), getdata5di16(), getdata5di32(), getdata5di64(), and getdata5di8().

Here is the caller graph for this function:

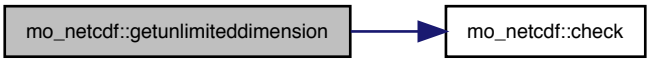


16.67.2.56 getunlimiteddimension()

```
type(ncdimension) function mo_netcdf::getunlimiteddimension (
            class(ncdataset), intent(in) self )
```

References check().

Here is the call graph for this function:

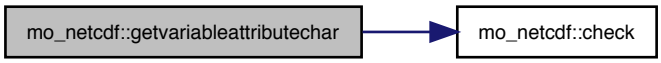


16.67.2.57 getvariableattributechar()

```
subroutine mo_netcdf::getvariableattributechar (
             class(ncvariable), intent(in) self,
             character(*), intent(in) name,
             character(*), intent(out) avalue )
```

References check().

Here is the call graph for this function:

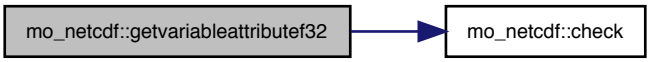


16.67.2.58 getvariableattributef32()

```
subroutine mo_netcdf::getvariableattributef32 (
             class(ncvariable), intent(in) self,
             character(*), intent(in) name,
             real(sp), intent(out) avalue )
```

References check().

Here is the call graph for this function:



16.67.2.59 getvariableattributef64()

```
subroutine mo_netcdf::getvariableattributef64 (
             class(ncvariable), intent(in) self,
             character(*), intent(in) name,
             real(dp), intent(out) avalue )
```

References check().

Here is the call graph for this function:

16.67.2.60 getvariableattributei16()

```
subroutine mo_netcdf::getvariableattributei16 (
             class(ncvariable), intent(in) self,
             character(*), intent(in) name,
             integer(i2), intent(out) avalue )
```

References check().

Here is the call graph for this function:

16.67.2.61 getvariableattributei32()

```
subroutine mo_netcdf::getvariableattributei32 (
             class(ncvariable), intent(in) self,
             character(*), intent(in) name,
             integer(i4), intent(out) avalue )
```

References check().

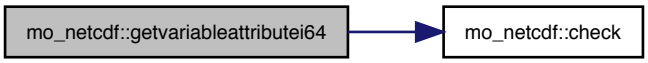
Here is the call graph for this function:

16.67.2.62 getvariableattributei64()

```
subroutine mo_netcdf::getvariableattributei64 (
             class(ncvariable), intent(in) self,
             character(*), intent(in) name,
             integer(i8), intent(out) avalue )
```

References check().

Here is the call graph for this function:



16.67.2.63 getvariableattributei8()

```
subroutine mo_netcdf::getvariableattributei8 (
             class(ncvariable), intent(in) self,
             character(*), intent(in) name,
             integer(i1), intent(out) avalue )
```

References check().

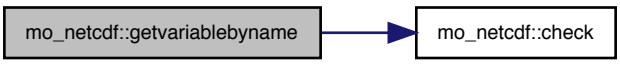
Here is the call graph for this function:

16.67.2.64 getvariablebyname()

```
type(ncvariable) function mo_netcdf::getvariablebyname (
             class(ncdataset), intent(in) self,
             character(*), intent(in) name )
```

References check().

Here is the call graph for this function:

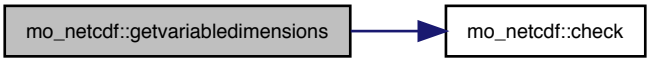


16.67.2.65 getvariabledimensions()

```
type(ncdimension) function, dimension(:), allocatable mo_netcdf::getvariabledimensions (
             class(ncvariable), intent(in) self )
```

References check().

Here is the call graph for this function:

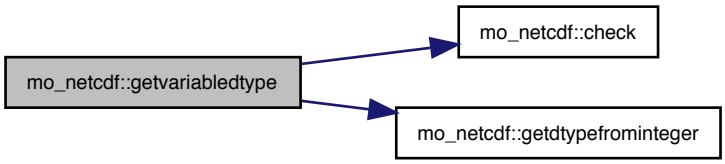


16.67.2.66 getvariabledtype()

```
character(3) function mo_netcdf::getvariabledtype (
             class(ncvariable), intent(in) self )
```

References check(), and getdtypefrominteger().

Here is the call graph for this function:



16.67.2.67 getvariablefillvaluef32()

```
subroutine mo_netcdf::getvariablefillvaluef32 (
             class(ncvariable), intent(in) self,
             real(sp), intent(out) fvalue )
```

16.67.2.68 getvariablefillvaluef64()

```
subroutine mo_netcdf::getvariablefillvaluef64 (
             class(ncvariable), intent(in) self,
             real(dp), intent(out) fvalue )
```

16.67.2.69 getvariablefillvaluei16()

```
subroutine mo_netcdf::getvariablefillvaluei16 (
             class(ncvariable), intent(in) self,
             integer(i2), intent(out) fvalue )
```

16.67.2.70 getvariablefillvaluei32()

```
subroutine mo_netcdf::getvariablefillvaluei32 (
             class(ncvariable), intent(in) self,
             integer(i4), intent(out) fvalue )
```

16.67.2.71 getvariablefillvaluei64()

```
subroutine mo_netcdf::getvariablefillvaluei64 (
             class(ncvariable), intent(in) self,
             integer(i8), intent(out) fvalue )
```

16.67.2.72 getvariablefillvaluei8()

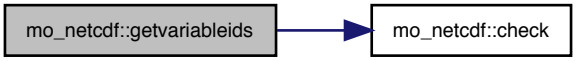
```
subroutine mo_netcdf::getvariablefillvaluei8 (
             class(ncvariable), intent(in) self,
             integer(i1), intent(out) fvalue )
```

16.67.2.73 getvariableids()

```
integer(i4) function, dimension(:), allocatable mo_netcdf::getvariableids (
             class(ncdataset), intent(in) self )
```

References check().

Here is the call graph for this function:

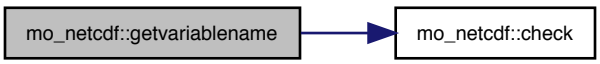


16.67.2.74 getvariablename()

```
character(len = 256) function mo_netcdf::getvariablename (
             class(ncvariable), intent(in) self )
```

References check().

Here is the call graph for this function:



16.67.2.75 getvariables()

```
type(ncvariable) function, dimension(:), allocatable mo_netcdf::getvariables (
             class(ncdataset), intent(in) self )
```

16.67.2.76 getvariableshape()

```
integer(i4) function, dimension(:), allocatable mo_netcdf::getvariableshape (
             class(ncvariable), intent(in) self )
```

16.67.2.77 hasattribute()

```
logical function mo_netcdf::hasattribute (
             class(ncvariable), intent(in) self,
             character(*), intent(in) name )
```

16.67.2.78 hasdimension()

```
logical function mo_netcdf::hasdimension (
             class(ncdataset), intent(in) self,
             character(*), intent(in) name )
```

16.67.2.79 hasvariable()

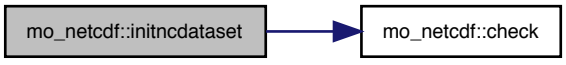
```
logical function mo_netcdf::hasvariable (
             class(ncdataset), intent(in) self,
             character(*), intent(in) name )
```

16.67.2.80 initncdataset()

```
subroutine mo_netcdf::initncdataset (
             class(ncdataset), intent(inout) self,
             character(*), intent(in) fname,
             character(1), intent(in) mode )
```

References check().

Here is the call graph for this function:



16.67.2.81 initncdimension()

```
subroutine mo_netcdf::initncdimension (
```

```
            class(ncdimension), intent(inout) self,
            integer(i4), intent(in) id,
            type(ncdataset), intent(in) parent )
```

16.67.2.82 initncvariable()

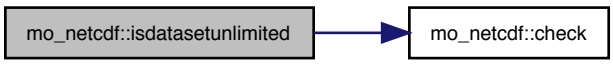
```
subroutine mo_netcdf::initncvariable (
            class(ncvariable), intent(inout) self,
            integer(i4), intent(in) id,
            type(ncdataset), intent(in) parent )
```

16.67.2.83 isdatasetunlimited()

```
logical function mo_netcdf::isdatasetunlimited (
            class(ncdataset), intent(in) self )
```

References check().

Here is the call graph for this function:



16.67.2.84 isunlimiteddimension()

```
logical function mo_netcdf::isunlimiteddimension (
            class(ncdimension), intent(in) self )
```

16.67.2.85 isunlimitedvariable()

```
logical function mo_netcdf::isunlimitedvariable (
            class(ncvariable), intent(in) self )
```

16.67.2.86 newncdataset()

```
type(ncdataset) function mo_netcdf::newncdataset (
            character(*), intent(in) fname,
            character(1), intent(in) mode )
```

16.67.2.87 newncdimension()

```
type(ncdimension) function mo_netcdf::newncdimension (
             integer(i4), intent(in) id,
             type(ncdataset), intent(in) parent )
```

16.67.2.88 newncvariable()

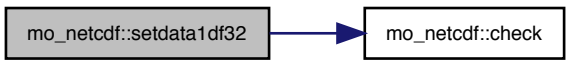
```
type(ncvariable) function mo_netcdf::newncvariable (
             integer(i4), intent(in) id,
             type(ncdataset), intent(in) parent )
```

16.67.2.89 setdata1df32()

```
subroutine mo_netcdf::setdata1df32 (
             class(ncvariable), intent(in) self,
             real(sp), dimension(:), intent(in) values,
             integer(i4), dimension(:), intent(in), optional start,
             integer(i4), dimension(:), intent(in), optional cnt,
             integer(i4), dimension(:), intent(in), optional stride,
             integer(i4), dimension(:), intent(in), optional map )
```

References check().

Here is the call graph for this function:

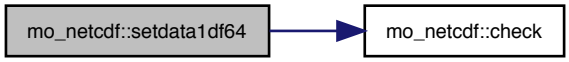


16.67.2.90 setdata1df64()

```
subroutine mo_netcdf::setdata1df64 (
             class(ncvariable), intent(in) self,
             real(dp), dimension(:), intent(in) values,
             integer(i4), dimension(:), intent(in), optional start,
             integer(i4), dimension(:), intent(in), optional cnt,
             integer(i4), dimension(:), intent(in), optional stride,
             integer(i4), dimension(:), intent(in), optional map )
```

References check().

Here is the call graph for this function:

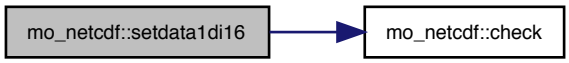


16.67.2.91 setdata1di16()

```
subroutine mo_netcdf::setdata1di16 (
            class(ncvariable), intent(in) self,
            integer(i2), dimension(:), intent(in) values,
            integer(i4), dimension(:), intent(in), optional start,
            integer(i4), dimension(:), intent(in), optional cnt,
            integer(i4), dimension(:), intent(in), optional stride,
            integer(i4), dimension(:), intent(in), optional map )
```

References check().

Here is the call graph for this function:

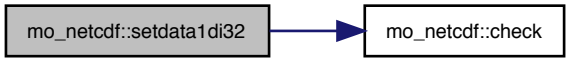


16.67.2.92 setdata1di32()

```
subroutine mo_netcdf::setdata1di32 (
            class(ncvariable), intent(in) self,
            integer(i4), dimension(:), intent(in) values,
            integer(i4), dimension(:), intent(in), optional start,
            integer(i4), dimension(:), intent(in), optional cnt,
            integer(i4), dimension(:), intent(in), optional stride,
            integer(i4), dimension(:), intent(in), optional map )
```

References check().

Here is the call graph for this function:

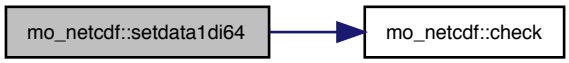


16.67.2.93 setdata1di64()

```
subroutine mo_netcdf::setdata1di64 (
             class(ncvariable), intent(in) self,
             integer(i8), dimension(:), intent(in) values,
             integer(i4), dimension(:), intent(in), optional start,
             integer(i4), dimension(:), intent(in), optional cnt,
             integer(i4), dimension(:), intent(in), optional stride,
             integer(i4), dimension(:), intent(in), optional map )
```

References check().

Here is the call graph for this function:

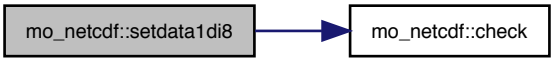


16.67.2.94 setdata1di8()

```
subroutine mo_netcdf::setdata1di8 (
             class(ncvariable), intent(in) self,
             integer(i1), dimension(:), intent(in) values,
             integer(i4), dimension(:), intent(in), optional start,
             integer(i4), dimension(:), intent(in), optional cnt,
             integer(i4), dimension(:), intent(in), optional stride,
             integer(i4), dimension(:), intent(in), optional map )
```

References check().

Here is the call graph for this function:

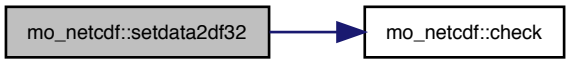


16.67.2.95 setdata2df32()

```
subroutine mo_netcdf::setdata2df32 (
             class(ncvariable), intent(in) self,
             real(sp), dimension(:, :), intent(in) values,
             integer(i4), dimension(:), intent(in), optional start,
             integer(i4), dimension(:), intent(in), optional cnt,
             integer(i4), dimension(:), intent(in), optional stride,
             integer(i4), dimension(:), intent(in), optional map )
```

References check().

Here is the call graph for this function:

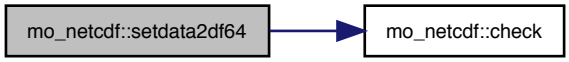


16.67.2.96 setdata2df64()

```
subroutine mo_netcdf::setdata2df64 (
             class(ncvariable), intent(in) self,
             real(dp), dimension(:, :), intent(in) values,
             integer(i4), dimension(:), intent(in), optional start,
             integer(i4), dimension(:), intent(in), optional cnt,
             integer(i4), dimension(:), intent(in), optional stride,
             integer(i4), dimension(:), intent(in), optional map )
```

References check().

Here is the call graph for this function:

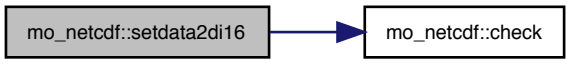


16.67.2.97 setdata2di16()

```
subroutine mo_netcdf::setdata2di16 (
             class(ncvariable), intent(in) self,
             integer(i2), dimension(:, :), intent(in) values,
             integer(i4), dimension(:), intent(in), optional start,
             integer(i4), dimension(:), intent(in), optional cnt,
             integer(i4), dimension(:), intent(in), optional stride,
             integer(i4), dimension(:), intent(in), optional map )
```

References check().

Here is the call graph for this function:

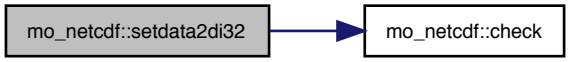


16.67.2.98 setdata2di32()

```
subroutine mo_netcdf::setdata2di32 (
             class(ncvariable), intent(in) self,
             integer(i4), dimension(:, :), intent(in) values,
             integer(i4), dimension(:), intent(in), optional start,
             integer(i4), dimension(:), intent(in), optional cnt,
             integer(i4), dimension(:), intent(in), optional stride,
             integer(i4), dimension(:), intent(in), optional map )
```

References check().

Here is the call graph for this function:

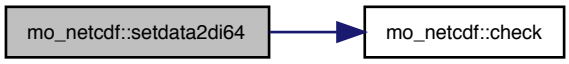


16.67.2.99 setdata2di64()

```
subroutine mo_netcdf::setdata2di64 (
             class(ncvariable), intent(in) self,
             integer(i8), dimension(:, :), intent(in) values,
             integer(i4), dimension(:), intent(in), optional start,
             integer(i4), dimension(:), intent(in), optional cnt,
             integer(i4), dimension(:), intent(in), optional stride,
             integer(i4), dimension(:), intent(in), optional map )
```

References check().

Here is the call graph for this function:

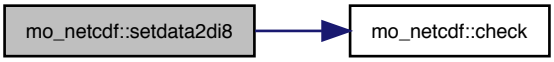


16.67.2.100 setdata2di8()

```
subroutine mo_netcdf::setdata2di8 (
             class(ncvariable), intent(in) self,
             integer(i1), dimension(:, :), intent(in) values,
             integer(i4), dimension(:), intent(in), optional start,
             integer(i4), dimension(:), intent(in), optional cnt,
             integer(i4), dimension(:), intent(in), optional stride,
             integer(i4), dimension(:), intent(in), optional map )
```

References check().

Here is the call graph for this function:

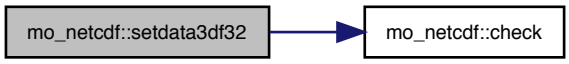


16.67.2.101 setdata3df32()

```
subroutine mo_netcdf::setdata3df32 (
             class(ncvariable), intent(in) self,
             real(sp), dimension(:, :, :), intent(in) values,
             integer(i4), dimension(:), intent(in), optional start,
             integer(i4), dimension(:), intent(in), optional cnt,
             integer(i4), dimension(:), intent(in), optional stride,
             integer(i4), dimension(:), intent(in), optional map )
```

References check().

Here is the call graph for this function:

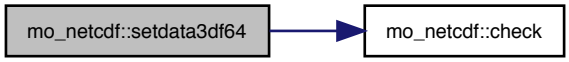


16.67.2.102 setdata3df64()

```
subroutine mo_netcdf::setdata3df64 (
             class(ncvariable), intent(in) self,
             real(dp), dimension(:, :, :), intent(in) values,
             integer(i4), dimension(:), intent(in), optional start,
             integer(i4), dimension(:), intent(in), optional cnt,
             integer(i4), dimension(:), intent(in), optional stride,
             integer(i4), dimension(:), intent(in), optional map )
```

References check().

Here is the call graph for this function:

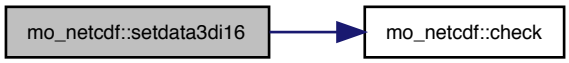


16.67.2.103 setdata3di16()

```
subroutine mo_netcdf::setdata3di16 (
             class(ncvariable), intent(in) self,
             integer(i2), dimension(:, :, :), intent(in) values,
             integer(i4), dimension(:), intent(in), optional start,
             integer(i4), dimension(:), intent(in), optional cnt,
             integer(i4), dimension(:), intent(in), optional stride,
             integer(i4), dimension(:), intent(in), optional map )
```

References check().

Here is the call graph for this function:

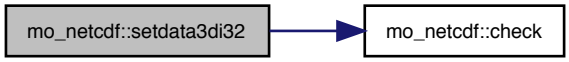


16.67.2.104 setdata3di32()

```
subroutine mo_netcdf::setdata3di32 (
             class(ncvariable), intent(in) self,
             integer(i4), dimension(:, :, :), intent(in) values,
             integer(i4), dimension(:), intent(in), optional start,
             integer(i4), dimension(:), intent(in), optional cnt,
             integer(i4), dimension(:), intent(in), optional stride,
             integer(i4), dimension(:), intent(in), optional map )
```

References check().

Here is the call graph for this function:

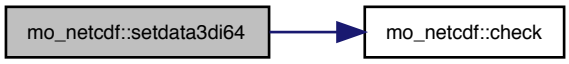


16.67.2.105 setdata3di64()

```
subroutine mo_netcdf::setdata3di64 (
             class(ncvariable), intent(in) self,
             integer(i8), dimension(:, :, :), intent(in) values,
             integer(i4), dimension(:), intent(in), optional start,
             integer(i4), dimension(:), intent(in), optional cnt,
             integer(i4), dimension(:), intent(in), optional stride,
             integer(i4), dimension(:), intent(in), optional map )
```

References check().

Here is the call graph for this function:

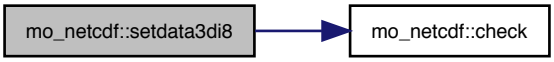


16.67.2.106 setdata3di8()

```
subroutine mo_netcdf::setdata3di8 (
             class(ncvariable), intent(in) self,
             integer(i1), dimension(:, :, :), intent(in) values,
             integer(i4), dimension(:), intent(in), optional start,
             integer(i4), dimension(:), intent(in), optional cnt,
             integer(i4), dimension(:), intent(in), optional stride,
             integer(i4), dimension(:), intent(in), optional map )
```

References check().

Here is the call graph for this function:

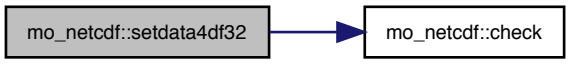


16.67.2.107 setdata4df32()

```
subroutine mo_netcdf::setdata4df32 (
             class(ncvariable), intent(in) self,
             real(sp), dimension(:, :, :, :), intent(in) values,
             integer(i4), dimension(:), intent(in), optional start,
             integer(i4), dimension(:), intent(in), optional cnt,
             integer(i4), dimension(:), intent(in), optional stride,
             integer(i4), dimension(:), intent(in), optional map )
```

References check().

Here is the call graph for this function:

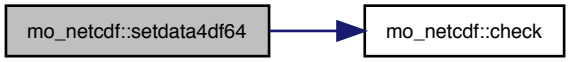


16.67.2.108 setdata4df64()

```
subroutine mo_netcdf::setdata4df64 (
             class(ncvariable), intent(in) self,
             real(dp), dimension(:, :, :, :), intent(in) values,
             integer(i4), dimension(:), intent(in), optional start,
             integer(i4), dimension(:), intent(in), optional cnt,
             integer(i4), dimension(:), intent(in), optional stride,
             integer(i4), dimension(:), intent(in), optional map )
```

References check().

Here is the call graph for this function:

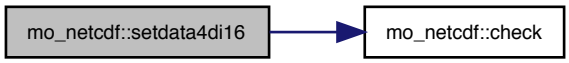


16.67.2.109 setdata4di16()

```
subroutine mo_netcdf::setdata4di16 (
             class(ncvariable), intent(in) self,
             integer(i2), dimension(:, :, :, :), intent(in) values,
             integer(i4), dimension(:), intent(in), optional start,
             integer(i4), dimension(:), intent(in), optional cnt,
             integer(i4), dimension(:), intent(in), optional stride,
             integer(i4), dimension(:), intent(in), optional map )
```

References check().

Here is the call graph for this function:

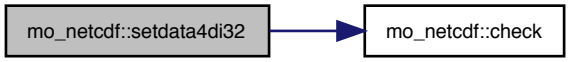


16.67.2.110 setdata4di32()

```
subroutine mo_netcdf::setdata4di32 (
             class(ncvariable), intent(in) self,
             integer(i4), dimension(:, :, :, :), intent(in) values,
             integer(i4), dimension(:), intent(in), optional start,
             integer(i4), dimension(:), intent(in), optional cnt,
             integer(i4), dimension(:), intent(in), optional stride,
             integer(i4), dimension(:), intent(in), optional map )
```

References check().

Here is the call graph for this function:

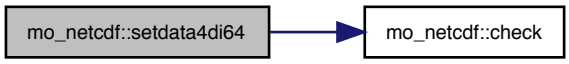


16.67.2.111 setdata4di64()

```
subroutine mo_netcdf::setdata4di64 (
             class(ncvariable), intent(in) self,
             integer(i8), dimension(:, :, :, :), intent(in) values,
             integer(i4), dimension(:), intent(in), optional start,
             integer(i4), dimension(:), intent(in), optional cnt,
             integer(i4), dimension(:), intent(in), optional stride,
             integer(i4), dimension(:), intent(in), optional map )
```

References check().

Here is the call graph for this function:

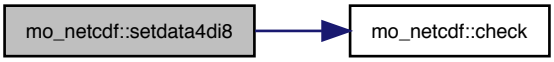


16.67.2.112 setdata4di8()

```
subroutine mo_netcdf::setdata4di8 (
             class(ncvariable), intent(in) self,
             integer(i1), dimension(:, :, :, :), intent(in) values,
             integer(i4), dimension(:), intent(in), optional start,
             integer(i4), dimension(:), intent(in), optional cnt,
             integer(i4), dimension(:), intent(in), optional stride,
             integer(i4), dimension(:), intent(in), optional map )
```

References check().

Here is the call graph for this function:

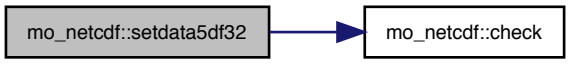


16.67.2.113 setdata5df32()

```
subroutine mo_netcdf::setdata5df32 (
             class(ncvariable), intent(in) self,
             real(sp), dimension(:, :, :, :, :), intent(in) values,
             integer(i4), dimension(:), intent(in), optional start,
             integer(i4), dimension(:), intent(in), optional cnt,
             integer(i4), dimension(:), intent(in), optional stride,
             integer(i4), dimension(:), intent(in), optional map )
```

References check().

Here is the call graph for this function:

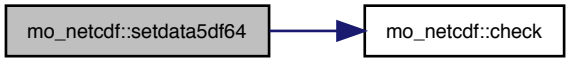


16.67.2.114 setdata5df64()

```
subroutine mo_netcdf::setdata5df64 (
             class(ncvariable), intent(in) self,
             real(dp), dimension(:, :, :, :, :), intent(in) values,
             integer(i4), dimension(:), intent(in), optional start,
             integer(i4), dimension(:), intent(in), optional cnt,
             integer(i4), dimension(:), intent(in), optional stride,
             integer(i4), dimension(:), intent(in), optional map )
```

References check().

Here is the call graph for this function:

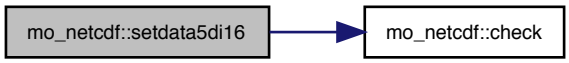


16.67.2.115 setdata5di16()

```
subroutine mo_netcdf::setdata5di16 (
             class(ncvariable), intent(in) self,
             integer(i2), dimension(:, :, :, :, :), intent(in) values,
             integer(i4), dimension(:), intent(in), optional start,
             integer(i4), dimension(:), intent(in), optional cnt,
             integer(i4), dimension(:), intent(in), optional stride,
             integer(i4), dimension(:), intent(in), optional map )
```

References check().

Here is the call graph for this function:

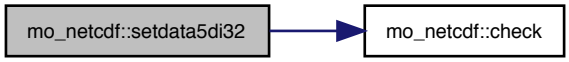


16.67.2.116 setdata5di32()

```
subroutine mo_netcdf::setdata5di32 (
             class(ncvariable), intent(in) self,
             integer(i4), dimension(:, :, :, :, :), intent(in) values,
             integer(i4), dimension(:), intent(in), optional start,
             integer(i4), dimension(:), intent(in), optional cnt,
             integer(i4), dimension(:), intent(in), optional stride,
             integer(i4), dimension(:), intent(in), optional map )
```

References check().

Here is the call graph for this function:

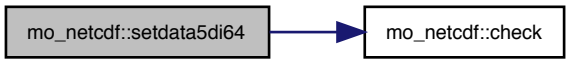


16.67.2.117 setdata5di64()

```
subroutine mo_netcdf::setdata5di64 (
             class(ncvariable), intent(in) self,
             integer(i8), dimension(:, :, :, :, :), intent(in) values,
             integer(i4), dimension(:), intent(in), optional start,
             integer(i4), dimension(:), intent(in), optional cnt,
             integer(i4), dimension(:), intent(in), optional stride,
             integer(i4), dimension(:), intent(in), optional map )
```

References check().

Here is the call graph for this function:



16.67.2.118 setdata5di8()

```
subroutine mo_netcdf::setdata5di8 (
             class(ncvariable), intent(in) self,
             integer(i1), dimension(:, :, :, :, :), intent(in) values,
             integer(i4), dimension(:), intent(in), optional start,
             integer(i4), dimension(:), intent(in), optional cnt,
             integer(i4), dimension(:), intent(in), optional stride,
             integer(i4), dimension(:), intent(in), optional map )
```

References check().

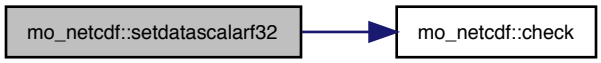
Here is the call graph for this function:

16.67.2.119 setdatascalarf32()

```
subroutine mo_netcdf::setdatascalarf32 (
             class(ncvariable), intent(in) self,
             real(sp), intent(in) values,
             integer(i4), dimension(:), intent(in), optional start )
```

References check().

Here is the call graph for this function:



16.67.2.120 setdatascalarf64()

```
subroutine mo_netcdf::setdatascalarf64 (
             class(ncvariable), intent(in) self,
             real(dp), intent(in) values,
             integer(i4), dimension(:), intent(in), optional start )
```

References check().

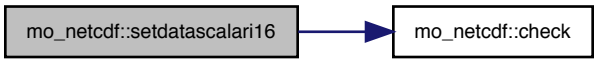
Here is the call graph for this function:

16.67.2.121 setdatascalari16()

```
subroutine mo_netcdf::setdatascalari16 (
            class(ncvariable), intent(in) self,
            integer(i2), intent(in) values,
            integer(i4), dimension(:), intent(in), optional start )
```

References check().

Here is the call graph for this function:

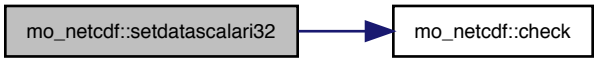


16.67.2.122 setdatascalari32()

```
subroutine mo_netcdf::setdatascalari32 (
            class(ncvariable), intent(in) self,
            integer(i4), intent(in) values,
            integer(i4), dimension(:), intent(in), optional start )
```

References check().

Here is the call graph for this function:



16.67.2.123 setdatascalari64()

```
subroutine mo_netcdf::setdatascalari64 (
            class(ncvariable), intent(in) self,
            integer(i8), intent(in) values,
            integer(i4), dimension(:), intent(in), optional start )
```

References check().

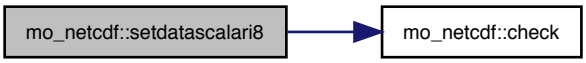
Here is the call graph for this function:

16.67.2.124 setdatascalari8()

```
subroutine mo_netcdf::setdatascalari8 (
             class(ncvariable), intent(in) self,
             integer(i1), intent(in) values,
             integer(i4), dimension(:), intent(in), optional start )
```

References check().

Here is the call graph for this function:

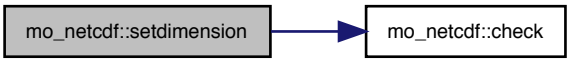


16.67.2.125 setdimension()

```
type(ncdimension) function mo_netcdf::setdimension (
             class(ncdataset), intent(in) self,
             character(*), intent(in) name,
             integer(i4), intent(in) length )
```

References check().

Here is the call graph for this function:



16.67.2.126 setglobalattributechar()

```
subroutine mo_netcdf::setglobalattributechar (
             class(ncdataset), intent(in) self,
             character(*), intent(in) name,
             character(*), intent(in) data )
```

References check().

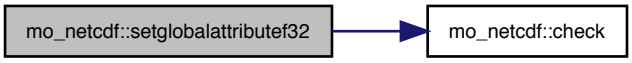
Here is the call graph for this function:

16.67.2.127 setglobalattributef32()

```
subroutine mo_netcdf::setglobalattributef32 (
             class(ncdataset), intent(in) self,
             character(*), intent(in) name,
             real(sp), intent(in) data )
```

References check().

Here is the call graph for this function:

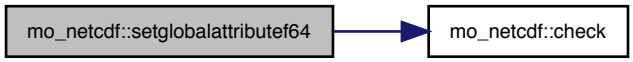


16.67.2.128 setglobalattributef64()

```
subroutine mo_netcdf::setglobalattributef64 (
             class(ncdataset), intent(in) self,
             character(*), intent(in) name,
             real(dp), intent(in) data )
```

References check().

Here is the call graph for this function:

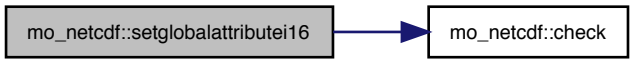


16.67.2.129 setglobalattributei16()

```
subroutine mo_netcdf::setglobalattributei16 (
             class(ncdataset), intent(in) self,
             character(*), intent(in) name,
             integer(i2), intent(in) data )
```

References check().

Here is the call graph for this function:

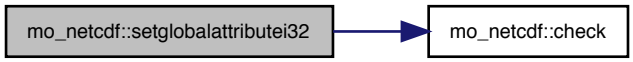


16.67.2.130 setglobalattributei32()

```
subroutine mo_netcdf::setglobalattributei32 (
             class(ncdataset), intent(in) self,
             character(*), intent(in) name,
             integer(i4), intent(in) data )
```

References check().

Here is the call graph for this function:

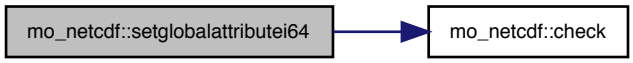


16.67.2.131 setglobalattributei64()

```
subroutine mo_netcdf::setglobalattributei64 (
             class(ncdataset), intent(in) self,
             character(*), intent(in) name,
             integer(i8), intent(in) data )
```

References check().

Here is the call graph for this function:



16.67.2.132 setglobalattributei8()

```
subroutine mo_netcdf::setglobalattributei8 (
             class(ncdataset), intent(in) self,
             character(*), intent(in) name,
             integer(i1), intent(in) data )
```

References check().

Here is the call graph for this function:

16.67.2.133 setvariableattributechar()

```
subroutine mo_netcdf::setvariableattributechar (
             class(ncvariable), intent(in) self,
             character(*), intent(in) name,
             character(*), intent(in) data )
```

References check().

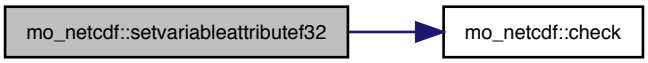
Here is the call graph for this function:

16.67.2.134 setvariableattributef32()

```
subroutine mo_netcdf::setvariableattributef32 (
             class(ncvariable), intent(in) self,
             character(*), intent(in) name,
             real(sp), intent(in) data )
```

References check().

Here is the call graph for this function:



16.67.2.135 setvariableattributef64()

```
subroutine mo_netcdf::setvariableattributef64 (
             class(ncvariable), intent(in) self,
             character(*), intent(in) name,
             real(dp), intent(in) data )
```

References check().

Here is the call graph for this function:

16.67.2.136 setvariableattributei16()

```
subroutine mo_netcdf::setvariableattributei16 (
             class(ncvariable), intent(in) self,
             character(*), intent(in) name,
             integer(i2), intent(in) data )
```

References check().

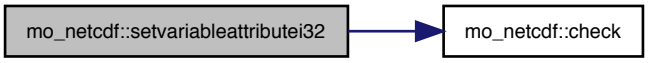
Here is the call graph for this function:

16.67.2.137 setvariableattributei32()

```
subroutine mo_netcdf::setvariableattributei32 (
             class(ncvariable), intent(in) self,
             character(*), intent(in) name,
             integer(i4), intent(in) data )
```

References check().

Here is the call graph for this function:

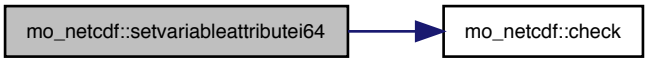


16.67.2.138 setvariableattributei64()

```
subroutine mo_netcdf::setvariableattributei64 (
             class(ncvariable), intent(in) self,
             character(*), intent(in) name,
             integer(i8), intent(in) data )
```

References check().

Here is the call graph for this function:



16.67.2.139 setvariableattributei8()

```
subroutine mo_netcdf::setvariableattributei8 (
             class(ncvariable), intent(in) self,
             character(*), intent(in) name,
             integer(i1), intent(in) data )
```

References check().

Here is the call graph for this function:

16.67.2.140 setvariablefillvaluef32()

```
subroutine mo_netcdf::setvariablefillvaluef32 (
             class(ncvariable), intent(in) self,
             real(sp), intent(in) fvalue )
```

16.67.2.141 setvariablefillvaluef64()

```
subroutine mo_netcdf::setvariablefillvaluef64 (
             class(ncvariable), intent(in) self,
             real(dp), intent(in) fvalue )
```

16.67.2.142 setvariablefillvaluei16()

```
subroutine mo_netcdf::setvariablefillvaluei16 (
```

```
            class(ncvariable), intent(in) self,
            integer(i2), intent(in) fvalue )
```

16.67.2.143 setvariablefillvaluei32()

```
subroutine mo_netcdf::setvariablefillvaluei32 (
            class(ncvariable), intent(in) self,
            integer(i4), intent(in) fvalue )
```

16.67.2.144 setvariablefillvaluei64()

```
subroutine mo_netcdf::setvariablefillvaluei64 (
            class(ncvariable), intent(in) self,
            integer(i8), intent(in) fvalue )
```

16.67.2.145 setvariablefillvaluei8()

```
subroutine mo_netcdf::setvariablefillvaluei8 (
            class(ncvariable), intent(in) self,
            integer(i1), intent(in) fvalue )
```

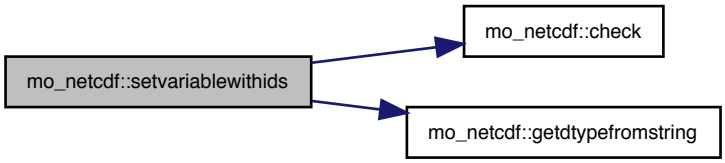
16.67.2.146 setvariablewithids()

```
type(ncvariable) function mo_netcdf::setvariablewithids (
            class(ncdataset), intent(in) self,
            character(*), intent(in) name,
            character(3), intent(in) dtype,
            integer(i4), dimension(:), intent(in) dimensions,
            logical, intent(in), optional contiguous,
            integer(i4), dimension(:), intent(in), optional chunksizes,
            integer(i4), intent(in), optional deflate_level,
            logical, intent(in), optional shuffle,
            logical, intent(in), optional fletcher32,
            integer(i4), intent(in), optional endianness,
            integer(i4), intent(in), optional cache_size,
            integer(i4), intent(in), optional cache_nelems,
            integer(i4), intent(in), optional cache_preemption )
```

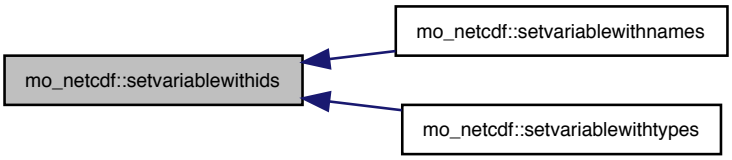
References check(), and getdtypefromstring().

Referenced by setvariablewithnames(), and setvariablewithtypes().

Here is the call graph for this function:



Here is the caller graph for this function:

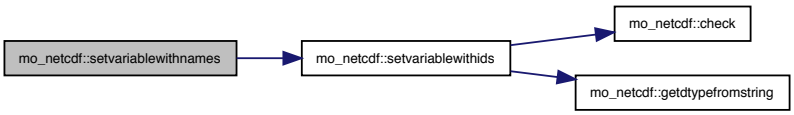


16.67.2.147 setvariablewithnames()

```
type(ncvariable) function mo_netcdf::setvariablewithnames (
             class(ncdataset), intent(in) self,
             character(*), intent(in) name,
             character(3), intent(in) dtype,
             character(*), dimension(:), intent(in) dimensions,
             logical, intent(in), optional contiguous,
             integer(i4), dimension(:), intent(in), optional chunksizes,
             integer(i4), intent(in), optional deflate_level,
             logical, intent(in), optional shuffle,
             logical, intent(in), optional fletcher32,
             integer(i4), intent(in), optional endianness,
             integer(i4), intent(in), optional cache_size,
             integer(i4), intent(in), optional cache_nelems,
             integer(i4), intent(in), optional cache_preemption )
```

References setvariablewithids().

Here is the call graph for this function:



16.67.2.148 setvariablewithtypes()

```
type(ncvariable) function mo_netcdf::setvariablewithtypes (
            class(ncdataset), intent(in) self,
            character(*), intent(in) name,
            character(3), intent(in) dtype,
            type(ncdimension), dimension(:), intent(in) dimensions,
            logical, intent(in), optional contiguous,
            integer(i4), dimension(:), intent(in), optional chunksizes,
            integer(i4), intent(in), optional deflate_level,
            logical, intent(in), optional shuffle,
            logical, intent(in), optional fletcher32,
            integer(i4), intent(in), optional endianness,
            integer(i4), intent(in), optional cache_size,
            integer(i4), intent(in), optional cache_nelems,
            integer(i4), intent(in), optional cache_preemption )
```

References setvariablewithids().

Here is the call graph for this function:

16.68 mo_neutrons Module Reference

Models to predict neutron intensities above soils.

Functions/Subroutines

- subroutine, public desiletsn0 (SoilMoisture, Horizons, N0, neutrons)

  Calculate neutrons from soil moisture in the first layer.
- subroutine, public cosmic (SoilMoisture, Horizons, params, neutron_integral_AFast, neutrons)

  Calculate neutrons from soil moisture in all layers.

- subroutine oldintegration (res, c)

  TODO: add description.
- subroutine, public tabularintegralafast (integral, maxC)

  Save approximation data for A_fast.
- subroutine approx_mon_int (res, f, c, xmin, xmax, eps, steps, fxmin, fxmax)

  TODO: add description.
- recursive subroutine approx_mon_int_steps (res, f, c, xmin, xmax, eps, steps, fxmin, fxmax)

  TODO: add description.
- recursive subroutine approx_mon_int_eps (res, f, c, xmin, xmax, eps, fxmin, fxmax)

  TODO: add description.
- subroutine lookupintegral (res, integral, c)

  TODO: add description.
- real(dp) function intgrandfast (c, phi)

  TODO: add description.

16.68.1 Detailed Description

Models to predict neutron intensities above soils.

The number of neutrons above the ground is directly related to the number soil water content in the ground, air, vegetation and/or snow. This module forward-models neutron abundance as a state variable for each cell.

Authors

Martin Schroen

Date

Mar 2015 THIS MODULE IS WORK IN PROGRESS, DO NOT USE FOR RESEARCH.

16.68.2 Function/Subroutine Documentation

16.68.2.1 approx_mon_int()

```
subroutine mo_neutrons::approx_mon_int (
             real(dp) res,
             real(dp), external f,
             real(dp), intent(in) c,
             real(dp), intent(in) xmin,
             real(dp), intent(in) xmax,
             real(dp), intent(in), optional eps,
             integer(i4), intent(in), optional steps,
             real(dp), intent(in), optional fxmin,
             real(dp), intent(in), optional fxmax )
```

TODO: add description.

TODO: add description

Parameters

in	*real(dp) :: c*	
in	*real(dp) :: xmin*	

Parameters

in	*real(dp) :: xmax*	
in	*real(dp), optional :: eps*	
in	*integer(i4), optional :: steps*	
in	*real(dp), optional :: fxmin*	
in	*real(dp), optional :: fxmax*	

Authors

Robert Schweppe

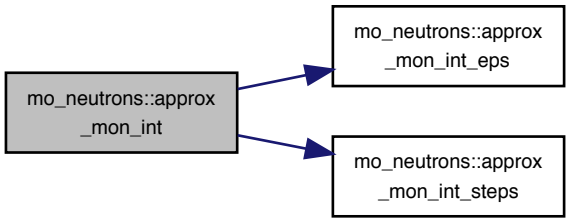
Date

Jun 2018

References approx_mon_int_eps(), and approx_mon_int_steps().

Referenced by tabularintegralafast().

Here is the call graph for this function:



Here is the caller graph for this function:

16.68.2.2 approx_mon_int_eps()

```
recursive subroutine mo_neutrons::approx_mon_int_eps (
            real(dp) res,
            real(dp), external f,
            real(dp), intent(in) c,
            real(dp), intent(in) xmin,
            real(dp), intent(in) xmax,
```

```
            real(dp), intent(in) eps,
            real(dp), intent(in) fxmin,
            real(dp), intent(in) fxmax )  [private]
```

TODO: add description.

TODO: add description

Parameters

in	*real(dp) :: c*	
in	*real(dp) :: xmin*	
in	*real(dp) :: xmax*	
in	*real(dp) :: eps*	
in	*real(dp) :: fxmin*	
in	*real(dp) :: fxmax*	

Authors

Robert Schweppe

Date

Jun 2018

Referenced by approx_mon_int().

Here is the caller graph for this function:

16.68.2.3 approx_mon_int_steps()

```
recursive subroutine mo_neutrons::approx_mon_int_steps (
            real(dp) res,
            real(dp), external f,
            real(dp), intent(in) c,
            real(dp), intent(in) xmin,
            real(dp), intent(in) xmax,
            real(dp), intent(in) eps,
            integer(i4), intent(in) steps,
            real(dp), intent(in) fxmin,
            real(dp), intent(in) fxmax )  [private]
```

TODO: add description.

TODO: add description

Parameters

in	*real(dp) :: c*	
in	*real(dp) :: xmin*	

Parameters

in	*real(dp) :: xmax*	
in	*real(dp) :: eps*	
in	*integer(i4) :: steps*	
in	*real(dp) :: fxmin*	
in	*real(dp) :: fxmax*	

Authors

Robert Schweppe

Date

Jun 2018

Referenced by approx_mon_int().

Here is the caller graph for this function:

16.68.2.4 cosmic()

```
subroutine, public mo_neutrons::cosmic (
             real(dp), dimension(:), intent(in) SoilMoisture,
             real(dp), dimension(:), intent(in) Horizons,
             real(dp), dimension(:), intent(in) params,
             real(dp), dimension(:), intent(in) neutron_integral_AFast,
             real(dp), intent(inout) neutrons )
```

Calculate neutrons from soil moisture in all layers.

Neutron counts above the ground (one value per cell in mHM) can be derived by a simplified physical neutron transport simulation. Fast cosmic-Ray neutrons are generated in the soil and attenuated differently in water and soil. The remaining neutrons that reached the surface relate to the profile of soil water content below. Variables like N, alpha and L3 are site-specific and need to be calibrated. ADDITIONAL INFORMATION COSMIC model based on Shuttleworth et al. 2013 Horizons(:) must not be zero. see supplementaries in literature J. Shuttleworth, R. Rosolem, M. Zreda, and T. Franz, The COsmic-ray Soil Moisture Interaction Code (COSMIC) for use in data assimilation, HESS, 17, 3205-3217, 2013, doi:10.5194/hess-17-3205-2013 Support and Code: http://cosmos.hwr.↩arizona.edu/Software/cosmic.html

Parameters

in	*real(dp), dimension(:) :: SoilMoisture*	Soil Moisture
in	*real(dp), dimension(:) :: Horizons*	Horizon depths
in	*real(dp), dimension(:) :: params*	! N0, N1, N2, alpha0, alpha1, L30, L31
in	*real(dp), dimension(:) :: neutron_integral_AFast*	Tabular for Int Approx
in, out	*real(dp) :: neutrons*	Neutron counts

Authors

Martin Schroen, originally written by Rafael Rosolem

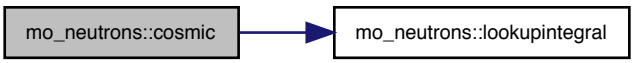
Date

Mar 2015

References mo_mhm_constants::cosmic_alpha, mo_mhm_constants::cosmic_bd, mo_mhm_constants::cosmic_↩
l1, mo_mhm_constants::cosmic_l2, mo_mhm_constants::cosmic_l3, mo_mhm_constants::cosmic_l4, mo_mhm↩
_constants::cosmic_n, mo_mhm_constants::cosmic_vwclat, mo_mhm_constants::h2odens, lookupintegral(), and mo_constants::pi_dp.

Referenced by mo_mhm::mhm().

Here is the call graph for this function:



Here is the caller graph for this function:

16.68.2.5 desiletsn0()

```
subroutine, public mo_neutrons::desiletsn0 (
             real(dp), dimension(:), intent(in) SoilMoisture,
             real(dp), dimension(:), intent(in) Horizons,
             real(dp), intent(in) N0,
             real(dp), intent(inout) neutrons )
```

Calculate neutrons from soil moisture in the first layer.

Using the N0-relation derived by Desilets, neutron counts above the ground (one value per cell in mHM) can be derived by a semi-empirical, semi-physical relation. The result depends on N0, the neutron counts for 0% soil mositure. This variable is site-specific and is a global parameter in mHM. N0 formula based on Desilets et al. 2010 Horizons(1) must not be zero. N0=1500cph, SoilMoisture(1,1)=700mm, Horizons(1)=200mm 1500∗(0.372+0.0808/(70mm/200mm + 0.115)) DesiletsN0 = 819cph Desilets, D., M. Zreda, and T. P. A. Ferre (2010), Nature's neutron probe: Land surface hydrology at an elusive scale with cosmic rays, WRR, 46, W11505, doi:10.1029/2009W↩
R008726.

Parameters

in	*real(dp), dimension(:) :: SoilMoisture*	Soil Moisture
in	*real(dp), dimension(:) :: Horizons*	Horizon depths
in	*real(dp) :: N0*	dry neutron counts
in,out	*real(dp) :: neutrons*	Neutron counts

Authors

Martin Schroen

Date

Mar 2015

References mo_mhm_constants::desilets_a0, mo_mhm_constants::desilets_a1, and mo_mhm_constants::desilets_a2.

Referenced by mo_mhm::mhm().

Here is the caller graph for this function:

16.68.2.6 intgrandfast()

```
real(dp) function mo_neutrons::intgrandfast (
            real(dp), intent(in) c,
            real(dp), intent(in) phi )
```

TODO: add description.

TODO: add description

Parameters

in	*real(dp) :: c*	
in	*real(dp) :: phi*	

Authors

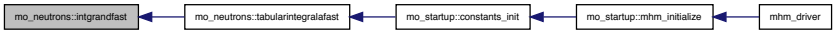
Robert Schweppe

Date

Jun 2018

Referenced by tabularintegralafast().

Here is the caller graph for this function:



16.68.2.7 lookupintegral()

```
subroutine mo_neutrons::lookupintegral (
            real(dp) res,
            real(dp), dimension(:), intent(in) integral,
            real(dp), intent(in) c )  [private]
```

TODO: add description.

TODO: add description

Parameters

in	*real(dp), dimension(:) :: integral*	
in	*real(dp) :: c*	

Authors

Robert Schweppe

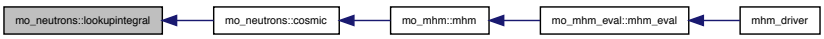
Date

Jun 2018

References mo_kind::i4, and mo_constants::pi_dp.

Referenced by cosmic().

Here is the caller graph for this function:



16.68.2.8 oldintegration()

```
subroutine mo_neutrons::oldintegration (
            real(dp) res,
            real(dp), intent(in) c )
```

TODO: add description.

TODO: add description

Parameters

in	*real(dp) :: c*	

Authors

Robert Schweppe

Date

Jun 2018

References mo_constants::pi_dp.

16.68.2.9 tabularintegralafast()

```
subroutine, public mo_neutrons::tabularintegralafast (
             real(dp), dimension(:)  integral,
             real(dp), intent(in) maxC )
```

Save approximation data for A_fast.

The COSMIC subroutine needs A_fast to be calculated. A_fast=int_{0}^{pi/2} exp(-Lambda_fast(z)/cos(phi) dphi) This subroutine stores data for intsize values for c=Lambda_fast(z) between 0 and maxC, and will be written into the global array variable neutron_integral_AFast. The calculation of the values is done with a very precise recursive approximation subroutine. That recursive subroutine should not be used inside the time, cells and layer loops, because it is slow. Inside the loops in the module COSMIC the tabular is used to estimate A_fast, if 0<c<maxC, otherwise the recursive approximation is used. TabularIntegralAFast: a tabular for calculations with splines intsize and maxC must be positive intsize=8000, maxC=20.0_dp see splines for example

Parameters

in	*real(dp) :: maxC*	! maximum value for A_fast

Authors

Maren Kaluza

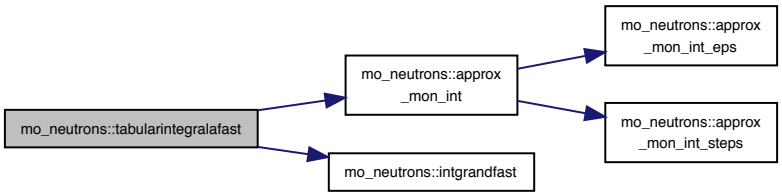
Date

Nov 2017

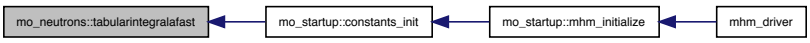
References approx_mon_int(), intgrandfast(), and mo_constants::pi_dp.

Referenced by mo_startup::constants_init().

Here is the call graph for this function:



Here is the caller graph for this function:



16.69 mo_nml Module Reference

Deal with namelist files.

Functions/Subroutines

- subroutine, public open_nml (file, unit, quiet)

  Open a namelist file.
- subroutine, public close_nml (unit)

  Close a namelist file.
- subroutine, public position_nml (name, unit, status, first)

  Position a namlist file.

Variables

- integer(i4), parameter, public positioned = 0

  Information: file pointer set to namelist group.
- integer(i4), parameter, public missing = 1

  Error: namelist group is missing.
- integer(i4), parameter, public length_error = 2

  Error: namelist group name too long.
- integer(i4), parameter, public read_error = 3

  Error occured during read of namelist file.
- integer, save, public nunitnml = -1

16.69.1 Detailed Description

Deal with namelist files.

This module provides routines to open, close and position namelist files.

Authors

Matthias Cuntz

Date

Jan 2011

16.69.2 Function/Subroutine Documentation

16.69.2.1 close_nml()

```
subroutine, public mo_nml::close_nml (
             integer, intent(in), optional unit )
```

Close a namelist file.

Parameters

in	*integer, optional :: unit*	namelist unit

Author

Matthias Cuntz - modified from Echam5, (C) MPI-MET, Hamburg, Germany

Date

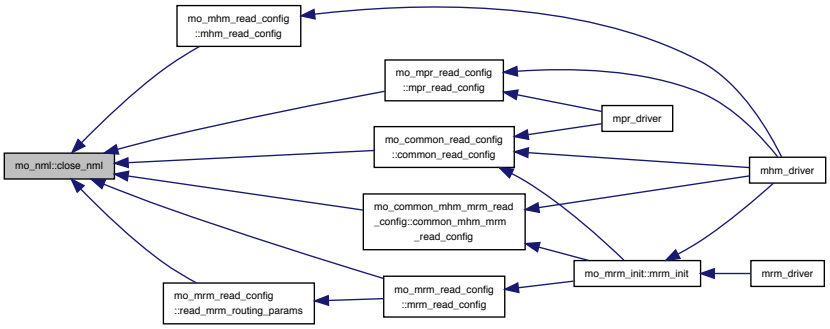
Dec 2011

References mo_finish::finish(), and nunitnml.

Referenced by mo_common_mhm_mrm_read_config::common_mhm_mrm_read_config(), mo_common_read_config::common_read_config(), mo_mhm_read_config::mhm_read_config(), mo_mpr_read_config::mpr_read_config(), mo_mrm_read_config::mrm_read_config(), and mo_mrm_read_config::read_mrm_routing_params().

Here is the call graph for this function:

Here is the caller graph for this function:



16.69.2.2 open_nml()

```
subroutine, public mo_nml::open_nml (
             character(len = *), intent(in) file,
             integer, intent(in) unit,
             logical, intent(in), optional quiet )
```

Open a namelist file.

Parameters

in	*character(len=*) :: file*	namelist filename
in	*integer :: unit*	namelist unit
in	*logical, optional :: quiet*	Be verbose or not (default: .true.) .true.: no messages .false.: write out messages

Author

Matthias Cuntz - modified from Echam5, (C) MPI-MET, Hamburg, Germany

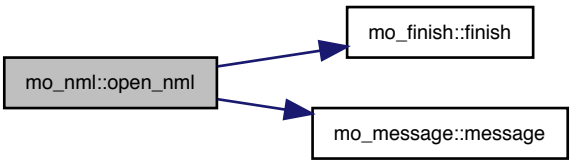
Date

Dec 2011

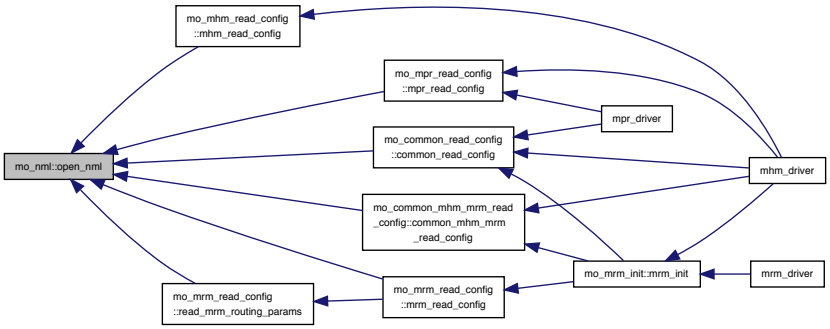
References mo_finish::finish(), mo_message::message(), mo_message::message_text, and nunitnml.

Referenced by mo_common_mhm_mrm_read_config::common_mhm_mrm_read_config(), mo_common_read_config::common_read_config(), mo_mhm_read_config::mhm_read_config(), mo_mpr_read_config::mpr_read_config(), mo_mrm_read_config::mrm_read_config(), and mo_mrm_read_config::read_mrm_routing_params().

Here is the call graph for this function:



Here is the caller graph for this function:



16.69.2.3 position_nml()

```
subroutine, public mo_nml::position_nml (
             character(len = *), intent(in) name,
             integer, intent(in), optional unit,
             integer(i4), intent(out), optional status,
             logical, intent(in), optional first )
```

Position a namlist file.

Position namelist file pointer for reading a new namelist next.
It positions the namelist file at the correct place for reading
namelist /name/ (case independent).

Parameters

in	*character(len=∗) :: name*	namelist name (case independent)
in	*integer, optional :: unit*	namelist unit (default: nunitnml)
in	*logical, optional :: first*	start search at beginning, i.e. rewind the namelist first (default: .true.) .true.: rewind .false.: continue search from current file pointer
out	*integer(i4), optional :: status*	Set on output to either of POSITIONED (0) - correct MISSING (1) - name not found LENGTH_ERROR (2) - namelist length longer then 256 characters READ_ERROR (3) - error while reading namelist file

Author

Matthias Cuntz - modified from Echam5, (C) MPI-MET, Hamburg, Germany

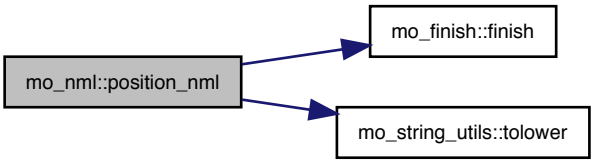
Date

Dec 2011

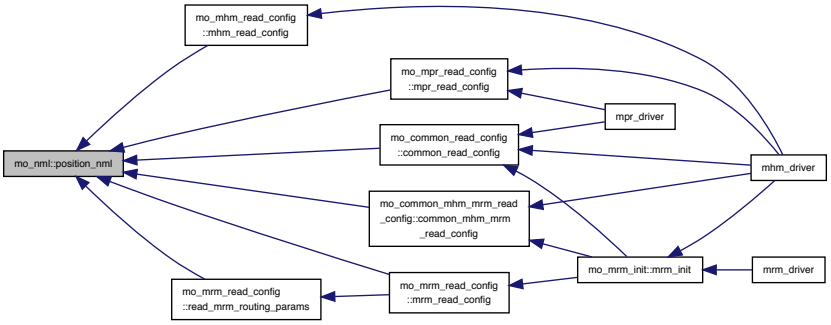
References mo_finish::finish(), length_error, mo_message::message_text, missing, nunitnml, positioned, read_↩ error, and mo_string_utils::tolower().

Referenced by mo_common_mhm_mrm_read_config::common_mhm_mrm_read_config(), mo_common_read↩ _config::common_read_config(), mo_mhm_read_config::mhm_read_config(), mo_mpr_read_config::mpr_read_↩ config(), mo_mrm_read_config::mrm_read_config(), and mo_mrm_read_config::read_mrm_routing_params().

Here is the call graph for this function:



Here is the caller graph for this function:



16.69.3 Variable Documentation

16.69.3.1 length_error

```
integer(i4), parameter, public mo_nml::length_error = 2
```

Error: namelist group name too long.

Referenced by position_nml().

16.69.3.2 missing

```
integer(i4), parameter, public mo_nml::missing = 1
```

Error: namelist group is missing.

Referenced by position_nml().

16.69.3.3 nunitnml

```
integer, save, public mo_nml::nunitnml = -1
```

Referenced by close_nml(), open_nml(), and position_nml().

16.69.3.4 positioned

```
integer(i4), parameter, public mo_nml::positioned = 0
```

Information: file pointer set to namelist group.

Referenced by position_nml().

16.69.3.5 read_error

```
integer(i4), parameter, public mo_nml::read_error = 3
```

Error occured during read of namelist file.

Referenced by position_nml().

16.70 mo_objective_function Module Reference

Objective Functions for Optimization of mHM.

Functions/Subroutines

- real(dp) function, public objective (parameterset, eval, arg1, arg2, arg3)

  Wrapper for objective functions.
- real(dp) function objective_sm_kge_catchment_avg (parameterset, eval)

  Objective function for soil moisture.
- real(dp) function objective_sm_corr (parameterset, eval)

  Objective function for soil moisture.
- real(dp) function objective_sm_pd (parameterset, eval)

  Objective function for soil moisture.
- real(dp) function objective_sm_sse_standard_score (parameterset, eval)

  Objective function for soil moisture.
- real(dp) function objective_kge_q_rmse_tws (parameterset, eval)

  Objective function of KGE for runoff and RMSE for basin_avg TWS (standarized scores)
- real(dp) function objective_neutrons_kge_catchment_avg (parameterset, eval)

  Objective function for neutrons.
- real(dp) function objective_et_kge_catchment_avg (parameterset, eval)

  Objective function for evpotranspirstion (et).
- real(dp) function objective_kge_q_sm_corr (parameterset, eval)

  Objective function of KGE for runoff and correlation for SM.
- real(dp) function objective_kge_q_et (parameterset, eval)

  Objective function of KGE for runoff and KGE for ET.
- real(dp) function objective_kge_q_rmse_et (parameterset, eval)

  Objective function of KGE for runoff and RMSE for basin_avg ET (standarized scores)
- subroutine extract_basin_avg_tws (basinId, tws, tws_sim, tws_obs, tws_obs_mask)

  extracts basin average tws data from global variables

16.70.1 Detailed Description

Objective Functions for Optimization of mHM.

This module provides a wrapper for several objective functions used to optimize mHM against various variables. If the objective is only regarding runoff move it to mRM/mo_mrm_objective_function_runoff.f90. If it contains besides runoff another variable like TWS implement it here. All the objective functions are supposed to be minimized! (10) SO: SM: 1.0 - KGE of catchment average soilmoisture (11) SO: SM: 1.0 - Pattern dissimilarity (PD) of spatially distributed soil moisture (12) SO: SM: Sum of squared errors (SSE) of spatially distributed standard score (normalization) of soil moisture (13) SO: SM: 1.0 - average temporal correlation of spatially distributed soil moisture (15) SO: Q + TWS: [1.0-KGE(Q)]∗RMSE(basin_avg_TWS) - objective function using Q and basin average (standard score) TWS (17) SO: N: 1.0 - KGE of spatio-temporal neutron data, catchment-average (27) SO: ET: 1.0 - KGE of catchment average evapotranspiration

Authors

Juliane Mai

Date

Dec 2012

16.70.2 Function/Subroutine Documentation

16.70.2.1 extract_basin_avg_tws()

```
subroutine mo_objective_function::extract_basin_avg_tws (
            integer(i4), intent(in) basinId,
            real(dp), dimension(:, :), intent(in) tws,
            real(dp), dimension(:), intent(out), allocatable tws_sim,
            real(dp), dimension(:), intent(out), allocatable tws_obs,
            logical, dimension(:), intent(out), allocatable tws_obs_mask )
```

extracts basin average tws data from global variables

extracts simulated and measured basin average tws from global variables, such that they overlay exactly. For measured tws, only the tws during the evaluation period are cut, not succeeding nodata values. For simulated tws, warming days as well as succeeding nodata values are neglected. see use in this module above

Parameters

in	*integer(i4) :: basinId*	current basin Id
in	*real(dp), dimension(:, :) :: tws*	simulated basin average tws
out	*real(dp), dimension(:) :: tws_sim*	aggregated simulated
out	*real(dp), dimension(:) :: tws_obs*	extracted measured
out	*logical, dimension(:) :: tws_obs_mask*	mask of no data values

Authors

Stephan Thober

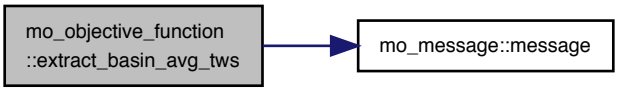
Date

Oct 2015

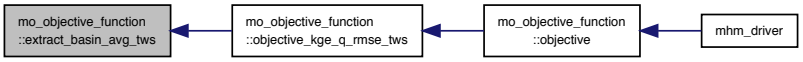
References mo_global_variables::basin_avg_tws_obs, mo_common_constants::eps_dp, mo_common_mhm_mrm_variables::evalper, mo_message::message(), mo_global_variables::nmeasperday_tws, mo_common_constants::nodata_dp, mo_common_mhm_mrm_variables::ntstepday, and mo_common_mhm_mrm_variables::warmingdays.

Referenced by objective_kge_q_rmse_tws().

Here is the call graph for this function:



Here is the caller graph for this function:



16.70.2.2 objective()

```
real(dp) function, public mo_objective_function::objective (
             real(dp), dimension(:), intent(in) parameterset,
             procedure(eval_interface), intent(in), pointer eval,
             real(dp), intent(in), optional arg1,
             real(dp), intent(out), optional arg2,
             real(dp), intent(out), optional arg3 )
```

Wrapper for objective functions.

The functions selects the objective function case defined in a namelist, i.e. the global variable *opti_function*. It return the objective function value for a specific parameter set.

Parameters

in	*REAL(dp), DIMENSION(:) :: parameterset*	
in	*procedure(eval_interface) :: eval*	
in	*real(dp), optional :: arg1*	
out	*real(dp), optional :: arg2*	
out	*real(dp), optional :: arg3*	

Returns

real(dp) :: objective — objective function value (which will be e.g. minimized by an optimization routine like DDS)

Authors

Juliane Mai

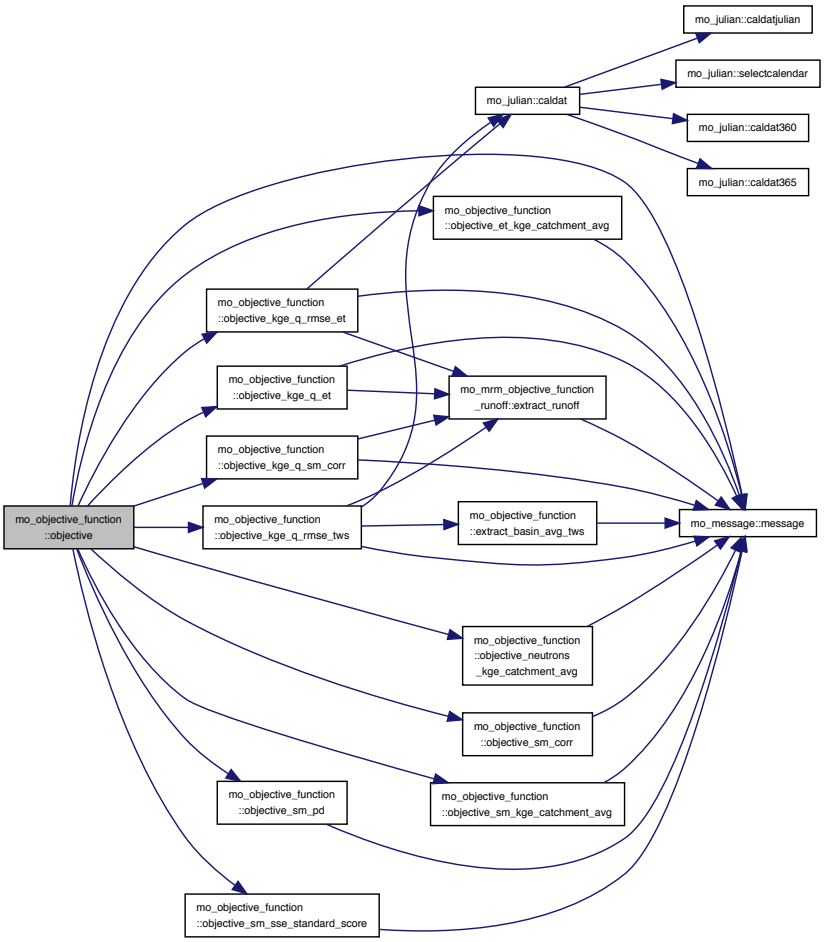
Date

Dec 2012

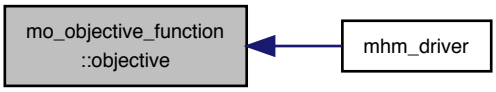
References mo_message::message(), mo_common_constants::nodata_dp, objective_et_kge_catchment_avg(), objective_kge_q_et(), objective_kge_q_rmse_et(), objective_kge_q_rmse_tws(), objective_kge_q_sm_corr(), objective_neutrons_kge_catchment_avg(), objective_sm_corr(), objective_sm_kge_catchment_avg(), objective_↩ sm_pd(), objective_sm_sse_standard_score(), and mo_common_mhm_mrm_variables::opti_function.

Referenced by mhm_driver().

Here is the call graph for this function:



Here is the caller graph for this function:



16.70.2.3 objective_et_kge_catchment_avg()

```
real(dp) function mo_objective_function::objective_et_kge_catchment_avg (
             real(dp), dimension(:), intent(in) parameterset,
             procedure(eval_interface), intent(in), pointer eval )
```

Objective function for evpotranspirstion (et).

The objective function only depends on a parameter vector. The model will be called with that parameter vector and the model output is subsequently compared to observed data. Therefore, the Kling-Gupta model efficiency KGE of the catchment average evapotranspiration (et) is calculated

$$KGE = 1.0 - \sqrt{((1-r)^2 + (1-\alpha)^2 + (1-\beta)^2)}$$

where r = Pearson product-moment correlation coefficient α = ratio of simulated mean to observed mean SM β = ratio of similated standard deviation to observed standard deviation is calculated and the objective function for a given basin i is

$$\phi_i = 1.0 - KGE_i$$

ϕ_i is the objective since we always apply minimization methods. The minimal value of ϕ_i is 0 for the optimal KGE of 1.0. Finally, the overall objective function value OF is estimated based on the power-6 norm to combine the ϕ_i from all basins N.

$$OF = \sqrt[6]{\sum((1.0 - KGE_i)/N)^6}.$$

The observed data L1_et, L1_et_mask are global in this module.

Parameters

in	*real(dp), dimension(:) :: parameterset*	
in	*procedure(eval_interface) :: eval*	

Returns

real(dp) :: objective_et_kge_catchment_avg — objective function value (which will be e.g. minimized by an optimization routine)

Authors

Johannes Brenner

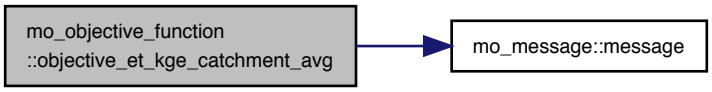
Date

Feb 2017

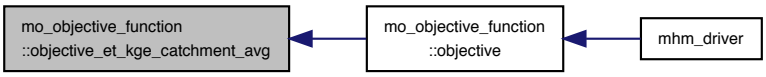
References mo_global_variables::l1_et, mo_global_variables::l1_et_mask, mo_common_variables::level1, mo_message::message(), mo_common_variables::nbasins, and mo_common_constants::nodata_dp.

Referenced by objective().

Here is the call graph for this function:



Here is the caller graph for this function:



16.70.2.4 objective_kge_q_et()

```
real(dp) function mo_objective_function::objective_kge_q_et (
            real(dp), dimension(:), intent(in) parameterset,
            procedure(eval_interface), intent(in), pointer eval )
```

Objective function of KGE for runoff and KGE for ET.

Objective function of KGE for runoff and KGE for ET. Further details can be found in the documentation of objective functions '14 - objective_multiple_gauges_kge_power6'.

Parameters

in	*real(dp), dimension(:) :: parameterset*	
in	*procedure(eval_interface) :: eval*	

Returns

real(dp) :: objective_kge_q_et — objective function value (which will be e.g. minimized by an optimization routine like DDS)

Authors

Johannes Brenner

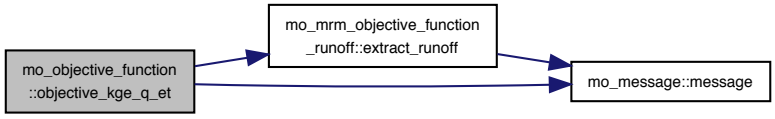
Date

July 2017

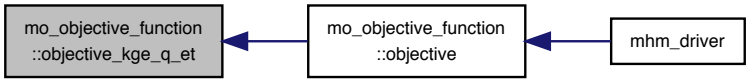
References mo_mrm_objective_function_runoff::extract_runoff(), mo_global_variables::l1_et, mo_global_↩ variables::l1_et_mask, mo_common_variables::level1, mo_message::message(), and mo_common_variables↩ ::nbasins.

Referenced by objective().

Here is the call graph for this function:



Here is the caller graph for this function:



16.70.2.5 objective_kge_q_rmse_et()

```
real(dp) function mo_objective_function::objective_kge_q_rmse_et (
            real(dp), dimension(:), intent(in) parameterset,
            procedure(eval_interface), intent(in), pointer eval )
```

Objective function of KGE for runoff and RMSE for basin_avg ET (standarized scores)

Objective function of KGE for runoff and RMSE for basin_avg ET (standarized scores)

Parameters

in	*real(dp), dimension(:) :: parameterset*	
in	*procedure(eval_interface) :: eval*	

Returns

real(dp) :: objective_kge_q_rmse_et — objective function value (which will be e.g. minimized by an optimization routine like DDS)

Authors

Johannes Brenner

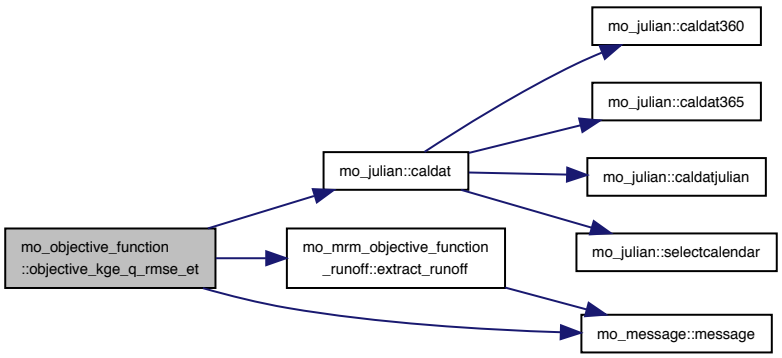
Date

July 2017

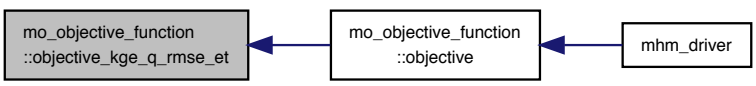
References mo_julian::caldat(), mo_common_constants::eps_dp, mo_common_mhm_mrm_variables::evalper, mo_mrm_objective_function_runoff::extract_runoff(), mo_global_variables::l1_et, mo_global_variables::l1_et_mask, mo_common_variables::level1, mo_message::message(), mo_common_variables::nbasins, mo_common_constants::nodata_dp, and mo_global_variables::timestep_et_input.

Referenced by objective().

Here is the call graph for this function:



Here is the caller graph for this function:



16.70.2.6 objective_kge_q_rmse_tws()

```
real(dp) function mo_objective_function::objective_kge_q_rmse_tws (
            real(dp), dimension(:), intent(in) parameterset,
            procedure(eval_interface), intent(in), pointer eval )
```

Objective function of KGE for runoff and RMSE for basin_avg TWS (standarized scores)

Objective function of KGE for runoff and RMSE for basin_avg TWS (standarized scores)

Parameters

in	*real(dp), dimension(:) :: parameterset*	
in	*procedure(eval_interface) :: eval*	

Returns

real(dp) :: objective_kge_q_rmse_tws — objective function value (which will be e.g. minimized by an optimization routine like DDS)

Authors

Oldrich Rakovec, Rohini Kumar

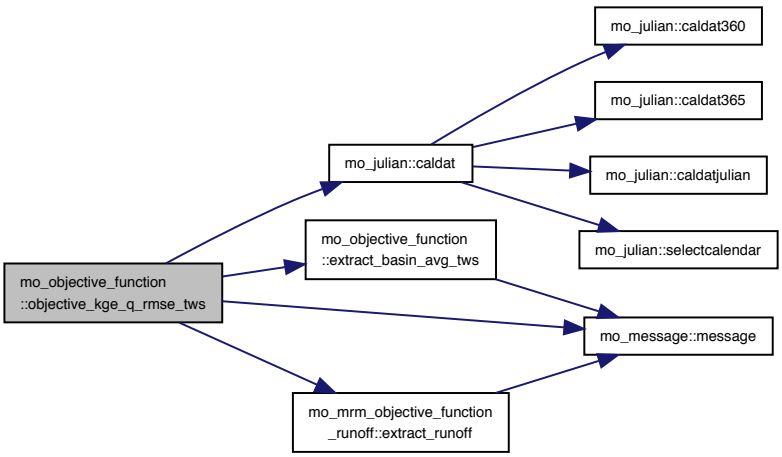
Date

Oct. 2015

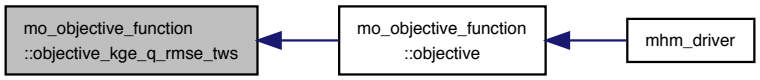
References mo_julian::caldat(), mo_common_constants::eps_dp, mo_common_mhm_mrm_variables::evalper, extract_basin_avg_tws(), mo_mrm_objective_function_runoff::extract_runoff(), mo_message::message(), mo_↩common_variables::nbasins, and mo_common_constants::nodata_dp.

Referenced by objective().

Here is the call graph for this function:



Here is the caller graph for this function:



16.70.2.7 objective_kge_q_sm_corr()

```
real(dp) function mo_objective_function::objective_kge_q_sm_corr (
             real(dp), dimension(:), intent(in) parameterset,
             procedure(eval_interface), intent(in), pointer eval )
```

Objective function of KGE for runoff and correlation for SM.

Objective function of KGE for runoff and SSE for soil moisture (standarized scores). Further details can be found in the documentation of objective functions '14 - objective_multiple_gauges_kge_power6' and '13 - objective_sm_corr'.

Parameters

in	*real(dp), dimension(:) :: parameterset*	
in	*procedure(eval_interface) :: eval*	

Returns

real(dp) :: objective_kge_q_sse_sm — objective function value (which will be e.g. minimized by an optimization routine like DDS)

Authors

Matthias Zink

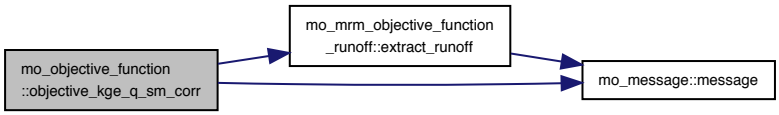
Date

Mar. 2017

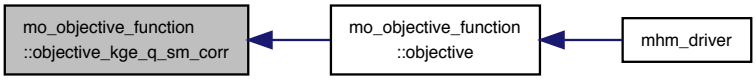
References mo_mrm_objective_function_runoff::extract_runoff(), mo_global_variables::l1_sm, mo_global_↩ variables::l1_sm_mask, mo_common_variables::level1, mo_message::message(), and mo_common_variables↩ ::nbasins.

Referenced by objective().

Here is the call graph for this function:



Here is the caller graph for this function:



16.70.2.8 objective_neutrons_kge_catchment_avg()

```
real(dp) function mo_objective_function::objective_neutrons_kge_catchment_avg (
             real(dp), dimension(:), intent(in) parameterset,
             procedure(eval_interface), intent(in), pointer eval )
```

Objective function for neutrons.

The objective function only depends on a parameter vector. The model will be called with that parameter vector and the model output is subsequently compared to observed data. Therefore, the Kling-Gupta model efficiency KGE of the catchment average neutrons (N) is calculated

$$KGE = 1.0 - \sqrt{((1-r)^2 + (1-\alpha)^2 + (1-\beta)^2)}$$

where r = Pearson product-moment CORRELATION coefficient α = ratio of simulated mean to observed mean SM β = ratio of similated standard deviation to observed standard deviation is calculated and the objective function for a given basin i is

$$\phi_i = 1.0 - KGE_i$$

ϕ_i is the objective since we always apply minimization methods. The minimal value of ϕ_i is 0 for the optimal KGE of 1.0. Finally, the overall objective function value OF is estimated based on the power-6 norm to combine the ϕ_i from all basins N.

$$OF = \sqrt[6]{\sum((1.0 - KGE_i)/N)^6}.$$

The observed data L1_neutronsdata, L1_neutronsdata_mask are global in this module.

Parameters

in	*real(dp), dimension(:) :: parameterset*	
in	*procedure(eval_interface) :: eval*	

Returns

real(dp) :: objective_neutrons_kge_catchment_avg — objective function value (which will be e.g. minimized by an optimization routine)

Authors

Martin Schroen

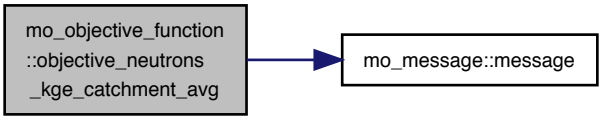
Date

Jun 2015

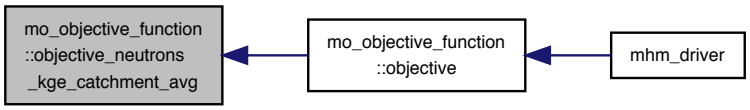
References mo_global_variables::l1_neutronsdata, mo_global_variables::l1_neutronsdata_mask, mo_common_variables::level1, mo_message::message(), mo_common_variables::nbasins, and mo_common_constants::nodata_dp.

Referenced by objective().

Here is the call graph for this function:



Here is the caller graph for this function:



16.70.2.9 objective_sm_corr()

```
real(dp) function mo_objective_function::objective_sm_corr (
             real(dp), dimension(:), intent(in) parameterset,
             procedure(eval_interface), intent(in), pointer eval )
```

Objective function for soil moisture.

The objective function only depends on a parameter vector. The model will be called with that parameter vector and the model output is subsequently compared to observed data. Therefore the Pearson correlation between observed and modeled soil moisture on each grid cell j is compared

$$r_j = r^2(SM^j_{obs}, SM^j_{sim})$$

where r^2 = Pearson correlation coefficient, SM_{obs} = observed soil moisture, SM_{sim} = simulated soil moisture. The observed data SM_{obs} are global in this module. The the correlation is spatially averaged as

$$\phi_i = \frac{1}{K} \cdot \sum_{j=1}^{K} r_j$$

where K denotes the number of valid cells in the study domain. Finally, the overall objective function value OF is estimated based on the power-6 norm to combine the ϕ_i from all basins N.

$$OF = \sqrt[6]{\sum((1.0 - \phi_i)/N)^6}.$$

The observed data L1_sm, L1_sm_mask are global in this module.

Parameters

in	*real(dp), dimension(:) :: parameterset*	
in	*procedure(eval_interface) :: eval*	

Returns

real(dp) :: objective_sm_corr — objective function value (which will be e.g. minimized by an optimization routine like DDS)

Authors

Matthias Zink

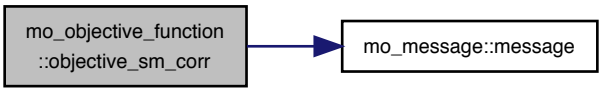
Date

March 2015

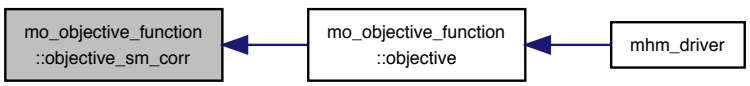
References mo_global_variables::l1_sm, mo_global_variables::l1_sm_mask, mo_common_variables::level1, mo↩_message::message(), and mo_common_variables::nbasins.

Referenced by objective().

Here is the call graph for this function:



Here is the caller graph for this function:



16.70.2.10 objective_sm_kge_catchment_avg()

```
real(dp) function mo_objective_function::objective_sm_kge_catchment_avg (
            real(dp), dimension(:), intent(in) parameterset,
            procedure(eval_interface), intent(in), pointer eval )
```

Objective function for soil moisture.

The objective function only depends on a parameter vector. The model will be called with that parameter vector and the model output is subsequently compared to observed data. Therefore, the Kling-Gupta model efficiency KGE of the catchment average soil mloisture (SM) is calculated

$$KGE = 1.0 - \sqrt{((1-r)^2 + (1-\alpha)^2 + (1-\beta)^2)}$$

where r = Pearson product-moment correlation coefficient α = ratio of simulated mean to observed mean SM β = ratio of similated standard deviation to observed standard deviation is calculated and the objective function for a given basin i is

$$\phi_i = 1.0 - KGE_i$$

ϕ_i is the objective since we always apply minimization methods. The minimal value of ϕ_i is 0 for the optimal KGE of 1.0. Finally, the overall objective function value OF is estimated based on the power-6 norm to combine the ϕ_i from all basins N.

$$OF = \sqrt[6]{\sum((1.0 - KGE_i)/N)^6}.$$

The observed data L1_sm, L1_sm_mask are global in this module.

Parameters

in	*real(dp), dimension(:) :: parameterset*	
in	*procedure(eval_interface) :: eval*	

Returns

real(dp) :: objective_sm_kge_catchment_avg — objective function value (which will be e.g. minimized by an optimization routine like DDS)

Authors

Matthias Zink

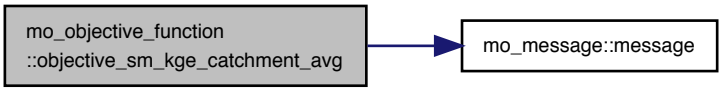
Date

May 2015

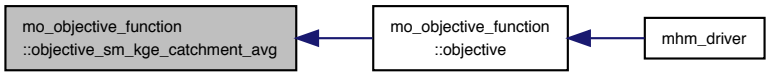
References mo_global_variables::l1_sm, mo_global_variables::l1_sm_mask, mo_common_variables::level1, mo↩_message::message(), mo_common_variables::nbasins, and mo_common_constants::nodata_dp.

Referenced by objective().

Here is the call graph for this function:



Here is the caller graph for this function:



16.70.2.11 objective_sm_pd()

```
real(dp) function mo_objective_function::objective_sm_pd (
             real(dp), dimension(:), intent(in) parameterset,
             procedure(eval_interface), intent(in), pointer eval )
```

Objective function for soil moisture.

The objective function only depends on a parameter vector. The model will be called with that parameter vector and the model output is subsequently compared to observed data. Therefore the Pattern Dissimilarity (PD) of observed and modeled soil moisture fields is calculated - aim: matching spatial patters

$$E(t) = PD\left(SM_{obs}(t), SM_{sim}(t)\right)$$

where PD = pattern dissimilarity function, SM_{obs} = observed soil moisture, SM_{sim} = simulated soil moisture. $E(t)$ = pattern dissimilarity at timestep t. The the pattern dissimilaity (E) is spatially averaged as

$$\phi_i = \frac{1}{T} \cdot \sum_{t=1}^{T} E_t$$

where T denotes the number of time steps. Finally, the overall objective function value OF is estimated based on the power-6 norm to combine the ϕ_i from all basins N.

$$OF = \sqrt[6]{\sum((1.0 - \phi_i)/N)^6}.$$

The observed data L1_sm, L1_sm_mask are global in this module. The observed data L1_sm, L1_sm_mask are global in this module.

Parameters

in	*real(dp), dimension(:) :: parameterset*	
in	*procedure(eval_interface) :: eval*	

Returns

real(dp) :: objecive_sm_pd — objective function value (which will be e.g. minimized by an optimization routine like DDS)

Authors

Matthias Zink

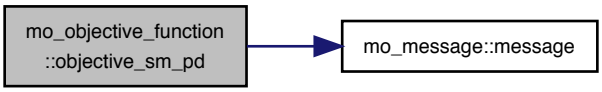
Date

May 2015

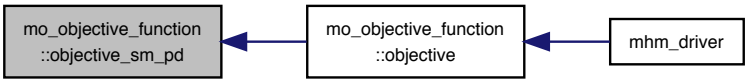
References mo_global_variables::l1_sm, mo_global_variables::l1_sm_mask, mo_common_variables::level1, mo_message::message(), mo_common_variables::nbasins, and mo_common_constants::nodata_dp.

Referenced by objective().

Here is the call graph for this function:



Here is the caller graph for this function:



16.70.2.12 objective_sm_sse_standard_score()

```
real(dp) function mo_objective_function::objective_sm_sse_standard_score (
            real(dp), dimension(:), intent(in) parameterset,
            procedure(eval_interface), intent(in), pointer eval )
```

Objective function for soil moisture.

The objective function only depends on a parameter vector. The model will be called with that parameter vector and the model output is subsequently compared to observed data. Therefore the sum of squared errors (SSE) of the standard score of observed and modeled soil moisture is calculated. The standard score or normalization (anomaly) make the objctive function bias insensitive and basically the dynamics of the soil moisture is tried to capture by this obejective function.

$$phi_i = \sum_{j=1}^{K} \{standard\_score(SM_{obs}(j)) - standard\_score(SM_{sim}(j))\}^2$$

where $standard\_score$ = standard score function, SM_{obs} = observed soil moisture, SM_{sim} = simulated soil moisture. K = valid elements in study domain. Finally, the overall objective function value OF is estimated based on the power-6 norm to combine the ϕ_i from all basins N.

$$OF = \sqrt[6]{\sum (\phi_i/N)^6}.$$

The observed data L1_sm, L1_sm_mask are global in this module.

Parameters

in	*real(dp), dimension(:) :: parameterset*	
in	*procedure(eval_interface) :: eval*	

Returns

real(dp) :: objective_sm_sse_standard_score — objective function value (which will be e.g. minimized by an optimization routine like DDS)

Authors

Matthias Zink

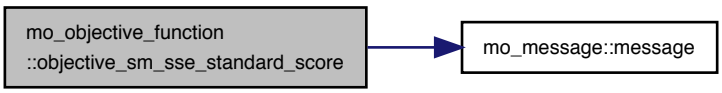
Date

March 2015

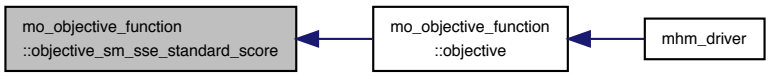
References mo_global_variables::l1_sm, mo_global_variables::l1_sm_mask, mo_common_variables::level1, mo↩_message::message(), and mo_common_variables::nbasins.

Referenced by objective().

Here is the call graph for this function:



Here is the caller graph for this function:



16.71 mo_optimization Module Reference

Wrapper subroutine for optimization against runoff and sm.

Functions/Subroutines

- subroutine, public optimization (eval, objective, dirConfigOut, funcBest, maskpara)

  Wrapper for optimization.

16.71.1 Detailed Description

Wrapper subroutine for optimization against runoff and sm.

This module provides a wrapper subroutine for optimization of mRM/mHM against runoff or soil moisture.

Authors

Stephan Thober

Date

Oct 2015

16.71.2 Function/Subroutine Documentation

16.71.2.1 optimization()

```
subroutine, public mo_optimization::optimization (
             procedure(eval_interface), intent(in), pointer eval,
             procedure(objective_interface), intent(in), pointer objective,
             character(len = *), intent(in) dirConfigOut,
             real(dp), intent(out) funcBest,
             logical, dimension(:), intent(out), allocatable maskpara )
```

Wrapper for optimization.

This subroutine selects the optimization defined in a namelist, i.e. the global variable *opti_method*. It return the objective function value for a specific parameter set.

Parameters

in	*procedure(eval_interface) :: eval*	
in	*procedure(objective_interface) :: objective*	- objective function used in the optimization
in	*character(len = *) :: dirConfigOut*	- directory where to write ascii output
out	*real(dp) :: funcbest*	- best objective function value obtained during optimization
out	*logical, dimension(:) :: maskpara*	true = parameter will be optimized = parameter(i,4) = 1 false = parameter will not be optimized = parameter(i,4) = 0

Authors

Matthias Cuntz, Luis Samaniego, Juliane Mai, Matthias Zink and Stephan Thober

Date

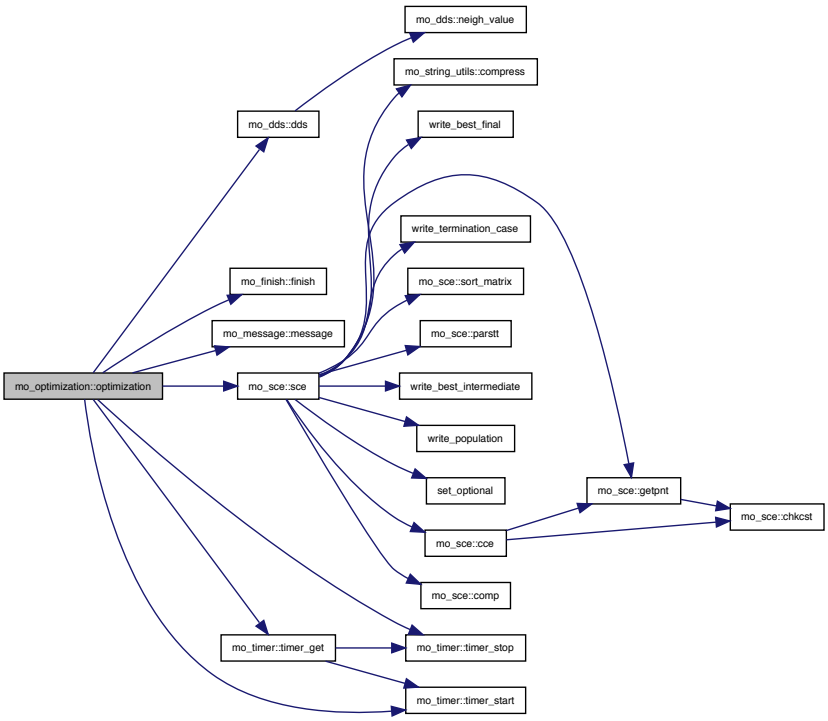
Oct 2015

References mo_dds::dds(), mo_common_mhm_mrm_variables::dds_r, mo_finish::finish(), mo_common_↩ variables::global_parameters, mo_kind::i4, mo_kind::i8, mo_common_mhm_mrm_variables::mcmc_error_↩ params, mo_common_mhm_mrm_variables::mcmc_opti, mo_message::message(), mo_common_mhm_mrm↩ _variables::niterations, mo_common_mhm_mrm_variables::opti_function, mo_common_mhm_mrm_variables↩ ::opti_method, mo_common_mhm_mrm_variables::optimize_restart, mo_common_mhm_mrm_variables::sa_↩ temp, mo_sce::sce(), mo_common_mhm_mrm_variables::sce_ngs, mo_common_mhm_mrm_variables::sce_npg,

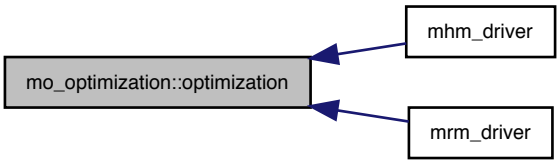
mo_common_mhm_mrm_variables::sce_nps, mo_common_mhm_mrm_variables::seed, mo_timer::timer_get(), mo_timer::timer_start(), and mo_timer::timer_stop().

Referenced by mhm_driver(), and mrm_driver().

Here is the call graph for this function:



Here is the caller graph for this function:



16.72 mo_optimization_utils Module Reference

Data Types

- interface eval_interface
- interface objective_interface

16.73 mo_orderpack Module Reference

Sort and ranking routines.

Data Types

- interface ctrper
- interface fndnth
- interface indmed
- interface indnth
- interface inspar
- interface inssor
- interface mrgref
- interface mrgrnk
- interface mulcnt
- interface nearless
- interface omedian
- interface rapknr
- interface refpar
- interface refsor
- interface rinpar
- interface rnkpar
- interface sort

  Unconditional ranking.
- interface sort_index
- interface uniinv
- interface unipar
- interface unirnk
- interface unista
- interface valmed
- interface valnth

Functions/Subroutines

- integer(i4) function, dimension(size(arr)) sort_index_dp (arr)
- integer(i4) function, dimension(size(arr)) sort_index_sp (arr)
- integer(i4) function, dimension(size(arr)) sort_index_i4 (arr)
- subroutine, private d_ctrper (XDONT, PCLS)
- subroutine, private r_ctrper (XDONT, PCLS)
- subroutine, private i_ctrper (XDONT, PCLS)
- real(kind=dp) function, private d_fndnth (XDONT, NORD)
- real(kind=sp) function, private r_fndnth (XDONT, NORD)
- integer(kind=i4) function, private i_fndnth (XDONT, NORD)
- subroutine, private d_indmed (XDONT, INDM)
- recursive subroutine, private d_med (XDATT, IDATT, ires_med)
- subroutine, private r_indmed (XDONT, INDM)
- recursive subroutine, private r_med (XDATT, IDATT, ires_med)
- subroutine, private i_indmed (XDONT, INDM)
- recursive subroutine, private i_med (XDATT, IDATT, ires_med)
- integer(kind=i4) function, private d_indnth (XDONT, NORD)
- integer(kind=i4) function, private r_indnth (XDONT, NORD)
- integer(kind=i4) function, private i_indnth (XDONT, NORD)
- subroutine, private d_inspar (XDONT, NORD)

- subroutine, private r_inspar (XDONT, NORD)
- subroutine, private i_inspar (XDONT, NORD)
- subroutine, private d_inssor (XDONT)
- subroutine, private r_inssor (XDONT)
- subroutine, private i_inssor (XDONT)
- real(kind=dp) function, private d_median (XDONT)
- real(kind=sp) function, private r_median (XDONT)
- integer(kind=i4) function, private i_median (XDONT)
- subroutine, private d_mrgref (XVALT, IRNGT)
- subroutine, private r_mrgref (XVALT, IRNGT)
- subroutine, private i_mrgref (XVALT, IRNGT)
- subroutine, private d_mrgrnk (XDONT, IRNGT)
- subroutine, private r_mrgrnk (XDONT, IRNGT)
- subroutine, private i_mrgrnk (XDONT, IRNGT)
- subroutine, private d_mulcnt (XDONT, IMULT)
- subroutine, private r_mulcnt (XDONT, IMULT)
- subroutine, private i_mulcnt (XDONT, IMULT)
- subroutine, private d_rapknr (XDONT, IRNGT, NORD)
- subroutine, private r_rapknr (XDONT, IRNGT, NORD)
- subroutine, private i_rapknr (XDONT, IRNGT, NORD)
- subroutine, private d_refpar (XDONT, IRNGT, NORD)
- subroutine, private r_refpar (XDONT, IRNGT, NORD)
- subroutine, private i_refpar (XDONT, IRNGT, NORD)
- subroutine, private d_refsor (XDONT)
- recursive subroutine, private d_subsor (XDONT, IDEB1, IFIN1)
- subroutine, private r_refsor (XDONT)
- recursive subroutine, private r_subsor (XDONT, IDEB1, IFIN1)
- subroutine, private i_refsor (XDONT)
- recursive subroutine, private i_subsor (XDONT, IDEB1, IFIN1)
- subroutine, private d_rinpar (XDONT, IRNGT, NORD)
- subroutine, private r_rinpar (XDONT, IRNGT, NORD)
- subroutine, private i_rinpar (XDONT, IRNGT, NORD)
- subroutine, private d_rnkpar (XDONT, IRNGT, NORD)
- subroutine, private r_rnkpar (XDONT, IRNGT, NORD)
- subroutine, private i_rnkpar (XDONT, IRNGT, NORD)
- subroutine, private d_uniinv (XDONT, IGOEST)
- subroutine, private r_uniinv (XDONT, IGOEST)
- subroutine, private i_uniinv (XDONT, IGOEST)
- real(kind=dp) function, private d_nearless (XVAL)
- real(kind=sp) function, private r_nearless (XVAL)
- integer(kind=i4) function, private i_nearless (XVAL)
- subroutine, private d_unipar (XDONT, IRNGT, NORD)
- subroutine, private r_unipar (XDONT, IRNGT, NORD)
- subroutine, private i_unipar (XDONT, IRNGT, NORD)
- subroutine, private d_unirnk (XVALT, IRNGT, NUNI)
- subroutine, private r_unirnk (XVALT, IRNGT, NUNI)
- subroutine, private i_unirnk (XVALT, IRNGT, NUNI)
- subroutine, private d_unista (XDONT, NUNI)
- subroutine, private r_unista (XDONT, NUNI)
- subroutine, private i_unista (XDONT, NUNI)
- recursive real(kind=dp) function, private d_valmed (XDONT)
- recursive real(kind=sp) function, private r_valmed (XDONT)
- recursive integer(kind=i4) function, private i_valmed (XDONT)
- real(kind=dp) function, private d_valnth (XDONT, NORD)
- real(kind=sp) function, private r_valnth (XDONT, NORD)
- integer(kind=i4) function, private i_valnth (XDONT, NORD)

Variables

- integer(kind=i4), dimension(:), allocatable, save idont

16.73.1 Detailed Description

Sort and ranking routines.

This module is the Orderpack 2.0 from Michel Olagnon. It provides order and unconditional, unique, and partial ranking, sorting, and permutation.

Authors

Michel Olagnon

Date

2000-2012

16.73.2 Function/Subroutine Documentation

16.73.2.1 d_ctrper()

```
subroutine, private mo_orderpack::d_ctrper (
             real(kind = dp), dimension (:), intent(inout) XDONT,
             real(kind = dp), intent(in) PCLS )  [private]
```

16.73.2.2 d_fndnth()

```
real(kind = dp) function, private mo_orderpack::d_fndnth (
             real(kind = dp), dimension (:), intent(in) XDONT,
             integer(kind = i4), intent(in) NORD )  [private]
```

16.73.2.3 d_indmed()

```
subroutine, private mo_orderpack::d_indmed (
             real(kind = dp), dimension (:), intent(in) XDONT,
             integer(kind = i4), intent(out) INDM )  [private]
```

References d_med(), and idont.

Here is the call graph for this function:

16.73.2.4 d_indnth()

```
integer(kind = i4) function, private mo_orderpack::d_indnth (
             real(kind = dp), dimension (:), intent(in) XDONT,
             integer(kind = i4), intent(in) NORD )  [private]
```

16.73.2.5 d_inspar()

```
subroutine, private mo_orderpack::d_inspar (
             real(kind = dp), dimension (:), intent(inout) XDONT,
             integer(kind = i4), intent(in) NORD )  [private]
```

16.73.2.6 d_inssor()

```
subroutine, private mo_orderpack::d_inssor (
             real(kind = dp), dimension (:), intent(inout) XDONT )  [private]
```

Referenced by d_refsor().

Here is the caller graph for this function:

16.73.2.7 d_med()

```
recursive subroutine, private mo_orderpack::d_med (
             real(kind = dp), dimension (:), intent(in) XDATT,
             integer(kind = i4), dimension (:), intent(in) IDATT,
             integer(kind = i4), intent(out) ires_med )  [private]
```

Referenced by d_indmed().

Here is the caller graph for this function:

16.73.2.8 d_median()

```
real(kind = dp) function, private mo_orderpack::d_median (
             real(kind = dp), dimension (:), intent(in) XDONT )  [private]
```

16.73.2.9 d_mrgref()

```
subroutine, private mo_orderpack::d_mrgref (
             real(kind = dp), dimension (:), intent(in) XVALT,
             integer(kind = i4), dimension (:), intent(out) IRNGT )  [private]
```

16.73.2.10 d_mrgrnk()

```
subroutine, private mo_orderpack::d_mrgrnk (
             real(kind = dp), dimension (:), intent(in) XDONT,
             integer(kind = i4), dimension (:), intent(out) IRNGT )  [private]
```

16.73.2.11 d_mulcnt()

```
subroutine, private mo_orderpack::d_mulcnt (
             real(kind = dp), dimension (:), intent(in) XDONT,
             integer(kind = i4), dimension (:), intent(out) IMULT )  [private]
```

16.73.2.12 d_nearless()

```
real(kind = dp) function, private mo_orderpack::d_nearless (
             real(kind = dp), intent(in) XVAL )  [private]
```

16.73.2.13 d_rapknr()

```
subroutine, private mo_orderpack::d_rapknr (
             real(kind = dp), dimension (:), intent(in) XDONT,
```

```
            integer(kind = i4), dimension (:), intent(out) IRNGT,
            integer(kind = i4), intent(in) NORD )  [private]
```

16.73.2.14 d_refpar()

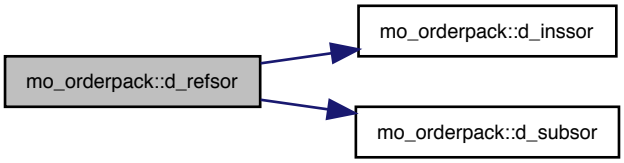
```
subroutine, private mo_orderpack::d_refpar (
            real(kind = dp), dimension (:), intent(in) XDONT,
            integer(kind = i4), dimension (:), intent(out) IRNGT,
            integer(kind = i4), intent(in) NORD )  [private]
```

16.73.2.15 d_refsor()

```
subroutine, private mo_orderpack::d_refsor (
            real(kind = dp), dimension (:), intent(inout) XDONT )  [private]
```

References d_inssor(), and d_subsor().

Here is the call graph for this function:



16.73.2.16 d_rinpar()

```
subroutine, private mo_orderpack::d_rinpar (
            real(kind = dp), dimension (:), intent(in) XDONT,
            integer(kind = i4), dimension (:), intent(out) IRNGT,
            integer(kind = i4), intent(in) NORD )  [private]
```

16.73.2.17 d_rnkpar()

```
subroutine, private mo_orderpack::d_rnkpar (
            real(kind = dp), dimension (:), intent(in) XDONT,
            integer(kind = i4), dimension (:), intent(out) IRNGT,
            integer(kind = i4), intent(in) NORD )  [private]
```

16.73.2.18 d_subsor()

```
recursive subroutine, private mo_orderpack::d_subsor (
             real(kind = dp), dimension (:), intent(inout) XDONT,
             integer(kind = i4), intent(in) IDEB1,
             integer(kind = i4), intent(in) IFIN1 )  [private]
```

Referenced by d_refsor().

Here is the caller graph for this function:

16.73.2.19 d_uniinv()

```
subroutine, private mo_orderpack::d_uniinv (
             real(kind = dp), dimension (:), intent(in) XDONT,
             integer(kind = i4), dimension (:), intent(out) IGOEST )  [private]
```

16.73.2.20 d_unipar()

```
subroutine, private mo_orderpack::d_unipar (
             real(kind = dp), dimension (:), intent(in) XDONT,
             integer(kind = i4), dimension (:), intent(out) IRNGT,
             integer(kind = i4), intent(inout) NORD )  [private]
```

16.73.2.21 d_unirnk()

```
subroutine, private mo_orderpack::d_unirnk (
             real(kind = dp), dimension (:), intent(in) XVALT,
             integer(kind = i4), dimension (:), intent(out) IRNGT,
             integer(kind = i4), intent(out) NUNI )  [private]
```

16.73.2.22 d_unista()

```
subroutine, private mo_orderpack::d_unista (
             real(kind = dp), dimension (:), intent(inout) XDONT,
             integer(kind = i4), intent(out) NUNI )  [private]
```

16.73.2.23 d_valmed()

```
recursive real(kind = dp) function, private mo_orderpack::d_valmed (
             real(kind = dp), dimension (:), intent(in) XDONT )  [private]
```

16.73.2.24 d_valnth()

```
real(kind = dp) function, private mo_orderpack::d_valnth (
             real(kind = dp), dimension (:), intent(in) XDONT,
             integer(kind = i4), intent(in) NORD )  [private]
```

16.73.2.25 i_ctrper()

```
subroutine, private mo_orderpack::i_ctrper (
             integer(kind = i4), dimension (:), intent(inout) XDONT,
             real(kind = sp), intent(in) PCLS )  [private]
```

16.73.2.26 i_fndnth()

```
integer(kind = i4) function, private mo_orderpack::i_fndnth (
             integer(kind = i4), dimension (:), intent(in) XDONT,
             integer(kind = i4), intent(in) NORD )  [private]
```

16.73.2.27 i_indmed()

```
subroutine, private mo_orderpack::i_indmed (
             integer(kind = i4), dimension (:), intent(in) XDONT,
             integer(kind = i4), intent(out) INDM )  [private]
```

References i_med(), and idont.

Here is the call graph for this function:

16.73.2.28 i_indnth()

```
integer(kind = i4) function, private mo_orderpack::i_indnth (
             integer(kind = i4), dimension (:), intent(in) XDONT,
```

```
            integer(kind = i4), intent(in) NORD )  [private]
```

16.73.2.29 i_inspar()

```
subroutine, private mo_orderpack::i_inspar (
            integer(kind = i4), dimension (:), intent(inout) XDONT,
            integer(kind = i4), intent(in) NORD )  [private]
```

16.73.2.30 i_inssor()

```
subroutine, private mo_orderpack::i_inssor (
            integer(kind = i4), dimension (:), intent(inout) XDONT )  [private]
```

Referenced by i_refsor().

Here is the caller graph for this function:

16.73.2.31 i_med()

```
recursive subroutine, private mo_orderpack::i_med (
            integer(kind = i4), dimension (:), intent(in) XDATT,
            integer(kind = i4), dimension (:), intent(in) IDATT,
            integer(kind = i4), intent(out) ires_med )  [private]
```

Referenced by i_indmed().

Here is the caller graph for this function:

16.73.2.32 i_median()

```
integer(kind = i4) function, private mo_orderpack::i_median (
             integer(kind = i4), dimension (:), intent(in) XDONT )  [private]
```

16.73.2.33 i_mrgref()

```
subroutine, private mo_orderpack::i_mrgref (
             integer(kind = i4), dimension (:), intent(in) XVALT,
             integer(kind = i4), dimension (:), intent(out) IRNGT )  [private]
```

16.73.2.34 i_mrgrnk()

```
subroutine, private mo_orderpack::i_mrgrnk (
             integer(kind = i4), dimension (:), intent(in) XDONT,
             integer(kind = i4), dimension (:), intent(out) IRNGT )  [private]
```

16.73.2.35 i_mulcnt()

```
subroutine, private mo_orderpack::i_mulcnt (
             integer(kind = i4), dimension (:), intent(in) XDONT,
             integer(kind = i4), dimension (:), intent(out) IMULT )  [private]
```

16.73.2.36 i_nearless()

```
integer(kind = i4) function, private mo_orderpack::i_nearless (
             integer(kind = i4), intent(in) XVAL )  [private]
```

16.73.2.37 i_rapknr()

```
subroutine, private mo_orderpack::i_rapknr (
             integer(kind = i4), dimension (:), intent(in) XDONT,
             integer(kind = i4), dimension (:), intent(out) IRNGT,
             integer(kind = i4), intent(in) NORD )  [private]
```

16.73.2.38 i_refpar()

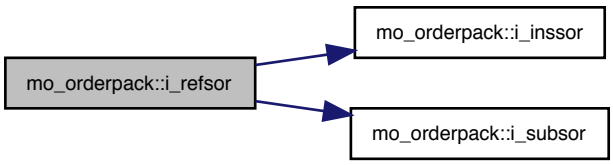
```
subroutine, private mo_orderpack::i_refpar (
             integer(kind = i4), dimension (:), intent(in) XDONT,
             integer(kind = i4), dimension (:), intent(out) IRNGT,
             integer(kind = i4), intent(in) NORD )  [private]
```

16.73.2.39 i_refsor()

```
subroutine, private mo_orderpack::i_refsor (
            integer(kind = i4), dimension (:), intent(inout) XDONT )  [private]
```

References i_inssor(), and i_subsor().

Here is the call graph for this function:



16.73.2.40 i_rinpar()

```
subroutine, private mo_orderpack::i_rinpar (
            integer(kind = i4), dimension (:), intent(in) XDONT,
            integer(kind = i4), dimension (:), intent(out) IRNGT,
            integer(kind = i4), intent(in) NORD )  [private]
```

16.73.2.41 i_rnkpar()

```
subroutine, private mo_orderpack::i_rnkpar (
            integer(kind = i4), dimension (:), intent(in) XDONT,
            integer(kind = i4), dimension (:), intent(out) IRNGT,
            integer(kind = i4), intent(in) NORD )  [private]
```

16.73.2.42 i_subsor()

```
recursive subroutine, private mo_orderpack::i_subsor (
            integer(kind = i4), dimension (:), intent(inout) XDONT,
            integer(kind = i4), intent(in) IDEB1,
            integer(kind = i4), intent(in) IFIN1 )  [private]
```

Referenced by i_refsor().

Here is the caller graph for this function:

16.73.2.43 i_uniinv()

```
subroutine, private mo_orderpack::i_uniinv (
             integer(kind = i4), dimension (:), intent(in) XDONT,
             integer(kind = i4), dimension (:), intent(out) IGOEST )  [private]
```

16.73.2.44 i_unipar()

```
subroutine, private mo_orderpack::i_unipar (
             integer(kind = i4), dimension (:), intent(in) XDONT,
             integer(kind = i4), dimension (:), intent(out) IRNGT,
             integer(kind = i4), intent(inout) NORD )  [private]
```

16.73.2.45 i_unirnk()

```
subroutine, private mo_orderpack::i_unirnk (
             integer(kind = i4), dimension (:), intent(in) XVALT,
             integer(kind = i4), dimension (:), intent(out) IRNGT,
             integer(kind = i4), intent(out) NUNI )  [private]
```

16.73.2.46 i_unista()

```
subroutine, private mo_orderpack::i_unista (
             integer(kind = i4), dimension (:), intent(inout) XDONT,
             integer(kind = i4), intent(out) NUNI )  [private]
```

16.73.2.47 i_valmed()

```
recursive integer(kind = i4) function, private mo_orderpack::i_valmed (
             integer(kind = i4), dimension (:), intent(in) XDONT )  [private]
```

16.73.2.48 i_valnth()

```
integer(kind = i4) function, private mo_orderpack::i_valnth (
             integer(kind = i4), dimension (:), intent(in) XDONT,
             integer(kind = i4), intent(in) NORD )  [private]
```

16.73.2.49 r_ctrper()

```
subroutine, private mo_orderpack::r_ctrper (
             real(kind = sp), dimension (:), intent(inout) XDONT,
             real(kind = sp), intent(in) PCLS )  [private]
```

16.73.2.50 r_fndnth()

```
real(kind = sp) function, private mo_orderpack::r_fndnth (
             real(kind = sp), dimension (:), intent(in) XDONT,
             integer(kind = i4), intent(in) NORD )  [private]
```

16.73.2.51 r_indmed()

```
subroutine, private mo_orderpack::r_indmed (
             real(kind = sp), dimension (:), intent(in) XDONT,
             integer(kind = i4), intent(out) INDM )  [private]
```

References idont, and r_med().

Here is the call graph for this function:

16.73.2.52 r_indnth()

```
integer(kind = i4) function, private mo_orderpack::r_indnth (
             real(kind = sp), dimension (:), intent(in) XDONT,
             integer(kind = i4), intent(in) NORD )  [private]
```

16.73.2.53 r_inspar()

```
subroutine, private mo_orderpack::r_inspar (
```

```
            real(kind = sp), dimension (:), intent(inout) XDONT,
            integer(kind = i4), intent(in) NORD )  [private]
```

16.73.2.54 r_inssor()

```
subroutine, private mo_orderpack::r_inssor (
            real(kind = sp), dimension (:), intent(inout) XDONT )  [private]
```

Referenced by r_refsor().

Here is the caller graph for this function:

16.73.2.55 r_med()

```
recursive subroutine, private mo_orderpack::r_med (
            real(kind = sp), dimension (:), intent(in) XDATT,
            integer(kind = i4), dimension (:), intent(in) IDATT,
            integer(kind = i4), intent(out) ires_med )  [private]
```

Referenced by r_indmed().

Here is the caller graph for this function:

16.73.2.56 r_median()

```
real(kind = sp) function, private mo_orderpack::r_median (
            real(kind = sp), dimension (:), intent(in) XDONT )  [private]
```

16.73.2.57 r_mrgref()

```
subroutine, private mo_orderpack::r_mrgref (
             real(kind = sp), dimension (:), intent(in) XVALT,
             integer(kind = i4), dimension (:), intent(out) IRNGT )  [private]
```

16.73.2.58 r_mrgrnk()

```
subroutine, private mo_orderpack::r_mrgrnk (
             real(kind = sp), dimension (:), intent(in) XDONT,
             integer(kind = i4), dimension (:), intent(out) IRNGT )  [private]
```

16.73.2.59 r_mulcnt()

```
subroutine, private mo_orderpack::r_mulcnt (
             real(kind = sp), dimension (:), intent(in) XDONT,
             integer(kind = i4), dimension (:), intent(out) IMULT )  [private]
```

16.73.2.60 r_nearless()

```
real(kind = sp) function, private mo_orderpack::r_nearless (
             real(kind = sp), intent(in) XVAL )  [private]
```

16.73.2.61 r_rapknr()

```
subroutine, private mo_orderpack::r_rapknr (
             real(kind = sp), dimension (:), intent(in) XDONT,
             integer(kind = i4), dimension (:), intent(out) IRNGT,
             integer(kind = i4), intent(in) NORD )  [private]
```

16.73.2.62 r_refpar()

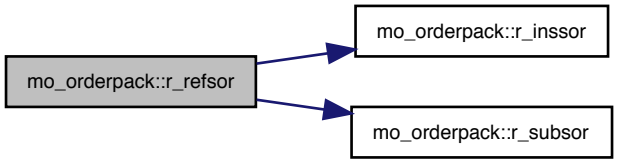
```
subroutine, private mo_orderpack::r_refpar (
             real(kind = sp), dimension (:), intent(in) XDONT,
             integer(kind = i4), dimension (:), intent(out) IRNGT,
             integer(kind = i4), intent(in) NORD )  [private]
```

16.73.2.63 r_refsor()

```
subroutine, private mo_orderpack::r_refsor (
             real(kind = sp), dimension (:), intent(inout) XDONT )  [private]
```

References r_inssor(), and r_subsor().

Here is the call graph for this function:



16.73.2.64 r_rinpar()

```
subroutine, private mo_orderpack::r_rinpar (
             real(kind = sp), dimension (:), intent(in) XDONT,
             integer(kind = i4), dimension (:), intent(out) IRNGT,
             integer(kind = i4), intent(in) NORD )  [private]
```

16.73.2.65 r_rnkpar()

```
subroutine, private mo_orderpack::r_rnkpar (
             real(kind = sp), dimension (:), intent(in) XDONT,
             integer(kind = i4), dimension (:), intent(out) IRNGT,
             integer(kind = i4), intent(in) NORD )  [private]
```

16.73.2.66 r_subsor()

```
recursive subroutine, private mo_orderpack::r_subsor (
             real(kind = sp), dimension (:), intent(inout) XDONT,
             integer(kind = i4), intent(in) IDEB1,
             integer(kind = i4), intent(in) IFIN1 )  [private]
```

Referenced by r_refsor().

Here is the caller graph for this function:

16.73.2.67 r_uniinv()

```
subroutine, private mo_orderpack::r_uniinv (
             real(kind = sp), dimension (:), intent(in) XDONT,
             integer(kind = i4), dimension (:), intent(out) IGOEST )  [private]
```

16.73.2.68 r_unipar()

```
subroutine, private mo_orderpack::r_unipar (
             real(kind = sp), dimension (:), intent(in) XDONT,
             integer(kind = i4), dimension (:), intent(out) IRNGT,
             integer(kind = i4), intent(inout) NORD )  [private]
```

16.73.2.69 r_unirnk()

```
subroutine, private mo_orderpack::r_unirnk (
             real(kind = sp), dimension (:), intent(in) XVALT,
             integer(kind = i4), dimension (:), intent(out) IRNGT,
             integer(kind = i4), intent(out) NUNI )  [private]
```

16.73.2.70 r_unista()

```
subroutine, private mo_orderpack::r_unista (
             real(kind = sp), dimension (:), intent(inout) XDONT,
             integer(kind = i4), intent(out) NUNI )  [private]
```

16.73.2.71 r_valmed()

```
recursive real(kind = sp) function, private mo_orderpack::r_valmed (
             real(kind = sp), dimension (:), intent(in) XDONT )  [private]
```

16.73.2.72 r_valnth()

```
real(kind = sp) function, private mo_orderpack::r_valnth (
             real(kind = sp), dimension (:), intent(in) XDONT,
             integer(kind = i4), intent(in) NORD )  [private]
```

16.73.2.73 sort_index_dp()

```
integer(i4) function, dimension(size(arr)) mo_orderpack::sort_index_dp (
             real(dp), dimension(:), intent(in) arr )  [private]
```

16.73.2.74 sort_index_i4()

```
integer(i4) function, dimension(size(arr)) mo_orderpack::sort_index_i4 (
             integer(i4), dimension(:), intent(in) arr )  [private]
```

16.73.2.75 sort_index_sp()

```
integer(i4) function, dimension(size(arr)) mo_orderpack::sort_index_sp (
             real(sp), dimension(:), intent(in) arr )  [private]
```

16.73.3 Variable Documentation

16.73.3.1 idont

```
integer(kind = i4), dimension(:), allocatable, save mo_orderpack::idont  [private]
```

Referenced by d_indmed(), i_indmed(), and r_indmed().

16.74 mo_percentile Module Reference

Data Types

- interface median
- interface n_element
- interface percentile
- interface qmedian

Functions/Subroutines

- real(dp) function median_dp (arrin, mask)
- real(sp) function median_sp (arrin, mask)
- real(dp) function n_element_dp (idat, n, mask, before, after, previous, next)
- real(sp) function n_element_sp (idat, n, mask, before, after, previous, next)
- real(dp) function percentile_0d_dp (arrin, k, mask, mode_in)
- real(sp) function percentile_0d_sp (arrin, k, mask, mode_in)
- real(dp) function, dimension(size(k)) percentile_1d_dp (arrin, k, mask, mode_in)
- real(sp) function, dimension(size(k)) percentile_1d_sp (arrin, k, mask, mode_in)
- real(dp) function qmedian_dp (dat)
- real(sp) function qmedian_sp (dat)

16.74.1 Function/Subroutine Documentation

16.74.1.1 median_dp()

```
real(dp) function mo_percentile::median_dp (
             real(dp), dimension(:), intent(in) arrin,
             logical, dimension(:), intent(in), optional mask )  [private]
```

16.74.1.2 median_sp()

```
real(sp) function mo_percentile::median_sp (
             real(sp), dimension(:), intent(in) arrin,
             logical, dimension(:), intent(in), optional mask )  [private]
```

16.74.1.3 n_element_dp()

```
real(dp) function mo_percentile::n_element_dp (
             real(dp), dimension(:), intent(in) idat,
             integer(i4), intent(in) n,
             logical, dimension(:), intent(in), optional mask,
             real(dp), intent(out), optional before,
             real(dp), intent(out), optional after,
             real(dp), intent(out), optional previous,
             real(dp), intent(out), optional next )  [private]
```

16.74.1.4 n_element_sp()

```
real(sp) function mo_percentile::n_element_sp (
             real(sp), dimension(:), intent(in) idat,
             integer(i4), intent(in) n,
             logical, dimension(:), intent(in), optional mask,
             real(sp), intent(out), optional before,
             real(sp), intent(out), optional after,
             real(sp), intent(out), optional previous,
             real(sp), intent(out), optional next )  [private]
```

16.74.1.5 percentile_0d_dp()

```
real(dp) function mo_percentile::percentile_0d_dp (
             real(dp), dimension(:), intent(in) arrin,
             real(dp), intent(in) k,
             logical, dimension(:), intent(in), optional mask,
             integer(i4), intent(in), optional mode_in )  [private]
```

References mo_kind::i4.

16.74.1.6 percentile_0d_sp()

```
real(sp) function mo_percentile::percentile_0d_sp (
             real(sp), dimension(:), intent(in) arrin,
```

```
            real(sp), intent(in) k,
            logical, dimension(:), intent(in), optional mask,
            integer(i4), intent(in), optional mode_in )  [private]
```

References mo_kind::i4.

16.74.1.7 percentile_1d_dp()

```
real(dp) function, dimension(size(k)) mo_percentile::percentile_1d_dp (
            real(dp), dimension(:), intent(in) arrin,
            real(dp), dimension(:), intent(in) k,
            logical, dimension(:), intent(in), optional mask,
            integer(i4), intent(in), optional mode_in )  [private]
```

References mo_kind::i4.

16.74.1.8 percentile_1d_sp()

```
real(sp) function, dimension(size(k)) mo_percentile::percentile_1d_sp (
            real(sp), dimension(:), intent(in) arrin,
            real(sp), dimension(:), intent(in) k,
            logical, dimension(:), intent(in), optional mask,
            integer(i4), intent(in), optional mode_in )  [private]
```

References mo_kind::i4.

16.74.1.9 qmedian_dp()

```
real(dp) function mo_percentile::qmedian_dp (
            real(dp), dimension(:), intent(inout) dat )  [private]
```

16.74.1.10 qmedian_sp()

```
real(sp) function mo_percentile::qmedian_sp (
            real(sp), dimension(:), intent(inout) dat )  [private]
```

16.75 mo_pet Module Reference

Module for calculating reference/potential evapotranspiration [mm s-1].

Functions/Subroutines

- elemental pure real(dp) function, public pet_hargreaves (HarSamCoeff, HarSamConst, tavg, tmax, tmin, latitude, doy)

  Reference Evapotranspiration after Hargreaves.
- elemental pure real(dp) function, public pet_priestly (PrieTayParam, Rn, tavg)

  Reference Evapotranspiration after Priestly-Taylor.

- elemental pure real(dp) function, public pet_penman (net_rad, tavg, act_vap_pressure, aerodyn_resistance, bulksurface_resistance, a_s, a_sh)

  Reference Evapotranspiration after Penman-Monteith.
- elemental pure real(dp) function, private extraterr_rad_approx (doy, latitude)

  Approximation of extraterrestrial radiation.
- elemental pure real(dp) function, private slope_satpressure (tavg)

  slope of saturation vapour pressure curve
- elemental pure real(dp) function, private sat_vap_pressure (tavg)

  calculation of the saturation vapour pressure

16.75.1 Detailed Description

Module for calculating reference/potential evapotranspiration [mm s-1].

This module calculates PET [mm/s] based on one of the methods

- Hargreaves-Samani (1982)
- Priestly-Taylor (1972)
- Penman-Monteith FAO (1998)

Authors

Matthias Zink, Christoph Schneider, Matthias Cuntz

Date

Apr 2014

16.75.2 Function/Subroutine Documentation

16.75.2.1 extraterr_rad_approx()

```
elemental pure real(dp) function, private mo_pet::extraterr_rad_approx (
            integer(i4), intent(in) doy,
            real(dp), intent(in) latitude )  [private]
```

Approximation of extraterrestrial radiation.

Approximation of extraterrestrial radiation at the top of the atmosphere R_a after Duffie and Beckman (1980). R_a is converted from $[J\ m^{-2}\ d^{-1}]$ in $[mm\ d^{-1}]$.

$$R_a = \frac{86400}{\pi \cdot \lambda} \cdot E_0 \cdot d_r \cdot (\omega \cdot \sin(latitude) \cdot \sin(\delta) + \cos(latitude) \cdot \cos(\delta) \cdot \sin(\omega)$$

where $E_0 = 1367\ J\ m^{-2}\ s^{-1}$ is the solar constant and It is dependent on the following sub equations: The relative distance Earth-Sun:

$$d_r = 1 + 0.033 \cdot \cos\left(\frac{2 \cdot \pi \cdot doy}{365}\right)$$

in which doy is the day of the year. The solar declination [radians] defined by

$$\delta = 0.4093 \cdot \sin\left(\frac{2 \cdot \pi \cdot doy}{365} - 1.405\right)$$

The sunset hour angle [radians]:

$$\omega = \arccos(-\tan(latitude) * \tan(\delta))$$

$< \lambda = 2.45 \cdot 10^6\ J\ m^{-2}\ mm^{-1}$ is the latent heat of vaporization.

Parameters

in	*integer(i4) :: doy*	day of year [-]
in	*real(dp) :: latitude*	latitude [rad]

Returns

real(dp) :: extraterr_rad_approx — extraterrestrial radiation approximation $[W\ m^{-2}]$

Authors

Matthias Zink

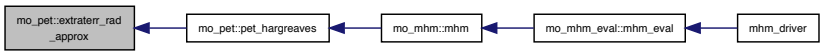
Date

Apr 2014

References mo_common_constants::daysecs, mo_mhm_constants::duffiedelta1, mo_mhm_constants::duffiedelta2, mo_mhm_constants::duffiedr, mo_constants::pi_d, mo_constants::solarconst_dp, mo_constants::specheatet_dp, mo_constants::twopi_d, and mo_common_constants::yeardays.

Referenced by pet_hargreaves().

Here is the caller graph for this function:



16.75.2.2 pet_hargreaves()

```
elemental pure real(dp) function, public mo_pet::pet_hargreaves (
             real(dp), intent(in) HarSamCoeff,
             real(dp), intent(in) HarSamConst,
             real(dp), intent(in) tavg,
             real(dp), intent(in) tmax,
             real(dp), intent(in) tmin,
             real(dp), intent(in) latitude,
             integer(i4), intent(in) doy )
```

Reference Evapotranspiration after Hargreaves.

Calculates the Reference Evapotranspiration $[mm\ d^{-1}]$ based on the Hargreaves-Samani (1982) model for a given cell by applying the equation

$$PET = HarSamCoeff * R_a * (T_{avg} + HarSamConst) * \sqrt{T_{max} - T_{min}}$$

where $R_a\ [W\ m^{-2}]$ is the incoming solar radiation and T_{avg}, T_{max} and $T_{min}\ [^{\circ}C]$ are the mean, maximum, and minimum daily temperatures at the given day, respectively.

Note

Hargreaves, G.H., and Samani, Z.A. (1982). "Estimating potential evapotranspiration." Tech. Note, J. Irrig. and drain. Engrg., ASCE, 108(3):225-230.

Parameters

in	*real(dp) :: HarSamCoeff*	coefficient of Hargreaves-Samani equation [-]
in	*real(dp) :: HarSamConst*	constant of Hargreaves-Samani equation [-]
in	*real(dp) :: tavg*	daily men temperature $[^{\circ}C]$
in	*real(dp) :: tmax*	daily maximum of temp $[^{\circ}C]$
in	*real(dp) :: tmin*	daily minimum of temp $[^{\circ}C]$
in	*real(dp) :: latitude*	latitude of the cell for Ra estimation $[radians]$
in	*integer(i4) :: doy*	day of year for Ra estimation

Returns

real(dp) :: pet_hargreaves — Hargreaves-Samani pot. evapotranspiration [mm s-1]

Authors

Matthias Zink

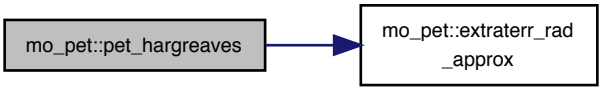
Date

Dec 2012

References mo_constants::deg2rad_dp, and extraterr_rad_approx().

Referenced by mo_mhm::mhm().

Here is the call graph for this function:



Here is the caller graph for this function:

16.75.2.3 pet_penman()

```
elemental pure real(dp) function, public mo_pet::pet_penman (
            real(dp), intent(in) net_rad,
```

```
real(dp), intent(in) tavg,
real(dp), intent(in) act_vap_pressure,
real(dp), intent(in) aerodyn_resistance,
real(dp), intent(in) bulksurface_resistance,
real(dp), intent(in) a_s,
real(dp), intent(in) a_sh )
```

Reference Evapotranspiration after Penman-Monteith.

Calculates the reference evapotranspiration $[mm\,d^{-1}]$ based on the Penman-Monteith model for every given cell by applying the equation

$$PET = \frac{1}{\lambda} \cdot \frac{\Delta \cdot R_n + \rho \cdot c_p \cdot (e_s - e) \cdot \frac{a_s h}{r_a}}{\Delta + \gamma \cdot \frac{a_s h}{a_s} \cdot \left(1 + \frac{r_s}{r_a}\right)}$$

where R_n $[W\ m^{-2}]$ is the net solar radiation, Δ $[kPa\ K^{-1}]$ is the slope of the saturation-vapour pressure curve, λ $[MJ\ kg^{-1}]$ is the latent heat of vaporization, $(e_s - e)$ $[kPa]$ is the vapour pressure deficit of the air, ρ $[kg\,m^{-3}]$ is the mean atmospheric density, $c_p = 1005.0\ J\ kg^{-1}\ K^{-1}$ is the specific heat of the air, $\gamma[kPa\ K^{-1}]$ is the psychrometric constant, $r_s[sm^{-1}]$ is the bulk canopy resistance, $r_a[sm^{-1}]$ is the aerodynamic resistance, $a_s[1]$ is the fraction of one-sided leaf area covered by stomata (1 if stomata are on one side only, 2 if they are on both sides) and $a_{sh}[-]$ is the fraction of projected area exchanging sensible heat with the air (2) Implementation refers to the so-called Penman-Montheith equation for transpiration. Adjusting the arguments a_{sh} and a_s we obtain the corrected MU equation (for details see Schymanski and Or, 2017). If $a_{sh} = 1 = a_s$ Penman-Montheith equation for transpiration is preserved. For reproducing characteristics of symmetrical amphistomatous leaves use $a_{sh} = 2 = a_s$, in which case the classic PM equation is only missing a factor of 2 in the nominator, as pointed out by Jarvis and McNaughton (1986, Eq. A9). These analytical solutions eliminated the non-linearity problem of the saturation vapour pressure curve, but they do not consider the dependency of the long-wave component of the soil surface or leaf energy balance (R_l) on soil or leaf temperature (T_l). We assume that net radiation equals the absorbed short-wave radiation, i.e. $R_N = R_s$ (p.79 in Monteith and Unsworth, 2013).

Parameters

in	*real(dp) :: net_rad*	net radiation $[Wm^{-2}]$
in	*real(dp) :: tavg*	average daily temperature $[°C]$
in	*real(dp) :: act_vap_pressure*	actual vapur pressure $[kPa]$
in	*real(dp) :: aerodyn_resistance*	aerodynmaical resistance $s\ m^{-1}$
in	*real(dp) :: bulksurface_resistance*	bulk surface resistance $s\ m^{-1}$
in	*real(dp) :: a_s*	fraction of one-sided leaf area covered by stomata 1
in	*real(dp) :: a_sh*	fraction of projected area exchanging sensible heat with the air 1

Returns

real(dp) :: pet_penman — Reference Evapotranspiration [mm s-1]

Authors

Matthias Zink

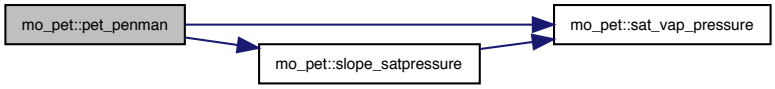
Date

Apr 2014

References mo_constants::cp0_dp, mo_common_constants::daysecs, mo_constants::psychro_dp, mo_↩ constants::rho0_dp, sat_vap_pressure(), slope_satpressure(), and mo_constants::specheatet_dp.

Referenced by mo_mhm::mhm().

Here is the call graph for this function:



Here is the caller graph for this function:

16.75.2.4 pet_priestly()

```
elemental pure real(dp) function, public mo_pet::pet_priestly (
             real(dp), intent(in) PrieTayParam,
             real(dp), intent(in) Rn,
             real(dp), intent(in) tavg )
```

Reference Evapotranspiration after Priestly-Taylor.

Calculates the Reference Evapotranspiration $[mm\ d^{-1}]$ based on the Priestly-Taylor (1972) model for every given cell by applying the equation

$$PET = \alpha * \frac{\Delta}{(\gamma + \Delta)} * R_n$$

where $R_n\ [W\ m^{-2}]$ is the net solar radiation $\Delta = f(T_{avg})$ is the slope of the saturation-vapour pressure curve and α is a emperical coefficient.

Parameters

in	*real(dp) :: PrieTayParam*	Priestley-Taylor coefficient $\alpha[-]$
in	*real(dp) :: Rn*	net solar radiation $[W\ m^{-2}]$
in	*real(dp) :: Tavg*	

Returns

real(dp) :: pet_priestly — Priestley-Taylor pot. evapotranspiration [mm s-1]

Authors

Matthias Zink

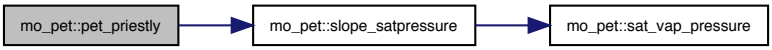
Date

Apr 2014

References mo_common_constants::daysecs, mo_constants::psychro_dp, slope_satpressure(), and mo_↩ constants::specheatet_dp.

Referenced by mo_mhm::mhm().

Here is the call graph for this function:



Here is the caller graph for this function:

16.75.2.5 sat_vap_pressure()

```
elemental pure real(dp) function, private mo_pet::sat_vap_pressure (
            real(dp), intent(in) tavg )  [private]
```

calculation of the saturation vapour pressure

Calculation of the saturation vapour pressure

$$e_s(T_a) = 0.6108 \cdot \exp\left(\frac{17.27 \cdot T_a}{T_a + 237.3}\right)$$

Parameters

in	*real(dp) :: tavg*	temperature [degC]

Returns

real(dp) :: sat_vap_pressure — saturation vapour pressure [kPa]

Authors

Matthias Zink

Date

Apr 2014

References mo_mhm_constants::tetens_c1, mo_mhm_constants::tetens_c2, and mo_mhm_constants::tetens_c3.

Referenced by pet_penman(), and slope_satpressure().

Here is the caller graph for this function:

16.75.2.6 slope_satpressure()

```
elemental pure real(dp) function, private mo_pet::slope_satpressure (
            real(dp), intent(in) tavg )  [private]
```

slope of saturation vapour pressure curve

slope of saturation vapour pressure curve after Tetens

$$\Delta = \frac{0.6108 * e_s(T_a)}{e^(2 \cdot \log(T_a + 237.3))}$$

Parameters

in	*real(dp) :: tavg*	average daily temperature $[°C]$

Returns

real(dp) :: slope_satpressure — slope of saturation vapour pressure curve $[kPa\ K-1]$

Authors

Matthias Zink

Date

Apr 2014

References sat_vap_pressure(), mo_mhm_constants::satpressureslope1, and mo_mhm_constants::tetens_c3.

Referenced by pet_penman(), and pet_priestly().

Here is the call graph for this function:

Here is the caller graph for this function:

16.76 mo_prepare_gridded_lai Module Reference

Prepare daily LAI fields (e.g., MODIS data) for mHM.

Functions/Subroutines

- subroutine, public prepare_gridded_daily_lai_data (iBasin, nrows, ncols, mask, LAIPer_iBasin)

  Prepare gridded daily LAI data.
- subroutine, public prepare_gridded_mean_monthly_lai_data (iBasin, nrows, ncols, mask)

  prepare_gridded_mean_monthly_LAI_data

16.76.1 Detailed Description

Prepare daily LAI fields (e.g., MODIS data) for mHM.

Prepare daily LAI fields(e.g., MODIS data) for mHM

Authors

John Craven & Rohini Kumar

Date

Aug 2013

16.76.2 Function/Subroutine Documentation

16.76.2.1 prepare_gridded_daily_lai_data()

```
subroutine, public mo_prepare_gridded_lai::prepare_gridded_daily_lai_data (
            integer(i4), intent(in) iBasin,
            integer(i4), intent(in) nrows,
            integer(i4), intent(in) ncols,
            logical, dimension(:, :), intent(in) mask,
            type(period), intent(in), optional LAIPer_iBasin )
```

Prepare gridded daily LAI data.

Prepare gridded daily LAI data at Level-0 (e.g., using MODIS datasets)

Parameters

in	*integer(i4) :: iBasin, nrows, ncols*	Basin Id
in	*integer(i4) :: iBasin, nrows, ncols*	Basin Id
in	*integer(i4) :: iBasin, nrows, ncols*	Basin Id
in	*logical, dimension(:, :) :: mask*	
in	*type(period), optional :: LAIPer_iBasin*	

Authors

John Craven & Rohini Kumar

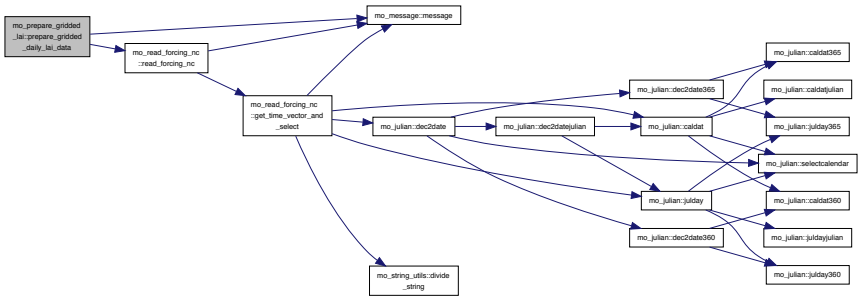
Date

Aug 2013

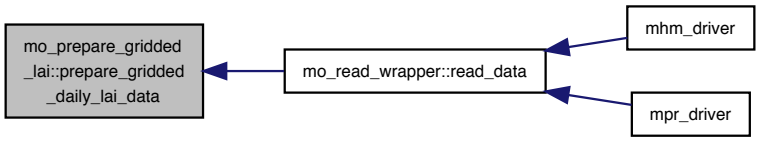
References mo_mpr_global_variables::dirgridded_lai, mo_mpr_global_variables::inputformat_gridded_lai, mo_mpr_global_variables::l0_gridded_lai, mo_message::message(), mo_mpr_global_variables::nlai, mo_read_forcing_nc::read_forcing_nc(), and mo_mpr_global_variables::timestep_lai_input.

Referenced by mo_read_wrapper::read_data().

Here is the call graph for this function:



Here is the caller graph for this function:



16.76.2.2 prepare_gridded_mean_monthly_lai_data()

```
subroutine, public mo_prepare_gridded_lai::prepare_gridded_mean_monthly_lai_data (
             integer(i4), intent(in) iBasin,
             integer(i4), intent(in) nrows,
             integer(i4), intent(in) ncols,
             logical, dimension(:, :), intent(in) mask )
```

prepare_gridded_mean_monthly_LAI_data

Long term mean monthly gridded LAI data at Level-0 (e.g., using MODIS datasets) The netcdf file should contain 12 (calender months) gridded fields of climatological LAI data at the input L0 data resolution.

Parameters

in	*integer(i4) :: iBasin, nrows, ncols*	Basin Id
in	*integer(i4) :: iBasin, nrows, ncols*	Basin Id
in	*integer(i4) :: iBasin, nrows, ncols*	Basin Id
in	*logical, dimension(:, :) :: mask*	

Authors

Rohini Kumar

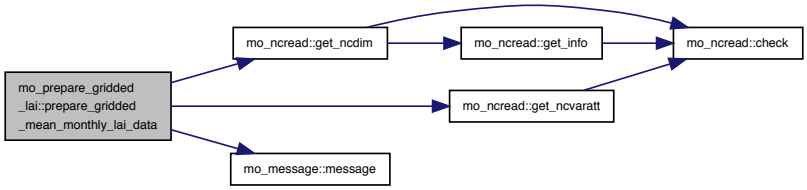
Date

Dec 2016

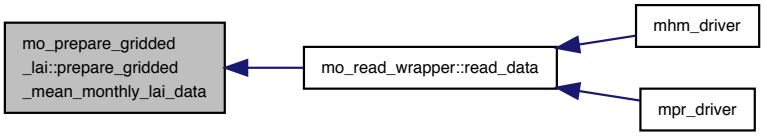
References mo_mpr_global_variables::dirgridded_lai, mo_ncread::get_ncdim(), mo_ncread::get_ncvaratt(), mo_↩ mpr_global_variables::l0_gridded_lai, mo_message::message(), and mo_mpr_global_variables::nlai.

Referenced by mo_read_wrapper::read_data().

Here is the call graph for this function:



Here is the caller graph for this function:



16.77 mo_read_forcing_nc Module Reference

Reads forcing input data.

Functions/Subroutines

- subroutine, public read_forcing_nc (folder, nRows, nCols, varName, mask, data, target_period, lower, upper, nctimestep, fileName, nocheck, maskout)

  Reads forcing input in NetCDF file format.
- subroutine, public read_const_forcing_nc (folder, nRows, nCols, varName, mask, data)
- subroutine, public read_weights_nc (folder, nRows, nCols, varName, data, mask, lower, upper, nocheck, maskout, fileName)

  Reads weights for meteo forcings input in NetCDF file format.
- subroutine get_time_vector_and_select (var, fname, inctimestep, time_start, time_cnt, target_period)

  TODO: add description.

16.77.1 Detailed Description

Reads forcing input data.

This module is to read forcing input data contained in netcdf files, e.g. temperature, precipitation, total_runoff, lai. Timesteps can be hourly, daily, monthly, and annual. The module provides a subroutine for NetCDF files only. First, the dimensions given are cross-checked with header.txt information. Second, the data of the specified period are read from the specified directory. If the optional lower and/or upper bound for the data values is given, the read data are checked for validity. The program is stopped if any value lies out of range.

Authors

Juliane Mai

Date

Dec 2012

16.77.2 Function/Subroutine Documentation

16.77.2.1 get_time_vector_and_select()

```
subroutine mo_read_forcing_nc::get_time_vector_and_select (
            type(ncvariable), intent(in) var,
            character(256), intent(in) fname,
            integer(i4), intent(in) inctimestep,
            integer(i4), intent(out) time_start,
            integer(i4), intent(out) time_cnt,
            type(period), intent(in), optional target_period )
```

TODO: add description.

TODO: add description ADDITIONAL INFORMATION get_time_vector_and_select Extract time vector in unit julian hours and get supposed time step in hours

Parameters

in	*type(NcVariable) :: var*	variable of interest
in	*character(256) :: fname*	fname of ncfile for error message
in	*integer(i4) :: inctimestep*	flag for requested time step
out	*integer(i4) :: time_start*	time_start index of time selection
out	*integer(i4) :: time_cnt*	time_count of indexes of time selection
in	*type(period), optional :: target_period*	reference period

Authors

Matthias Zink

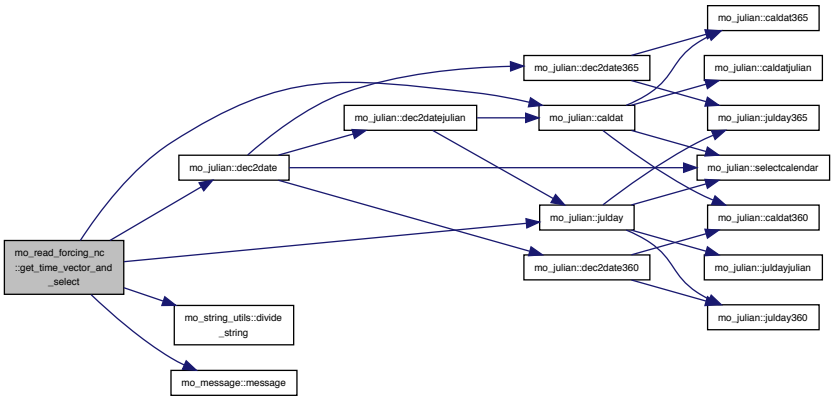
Date

Oct 2012

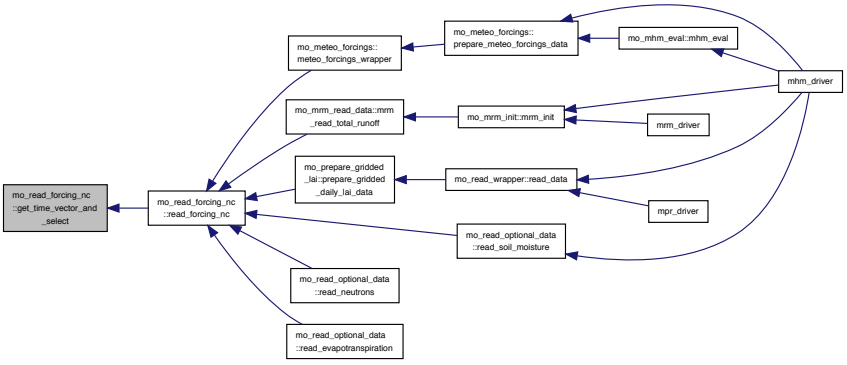
References mo_julian::caldat(), mo_constants::dayhours, mo_constants::daysecs, mo_julian::dec2date(), mo_↩string_utils::divide_string(), mo_kind::dp, mo_kind::i4, mo_kind::i8, mo_julian::julday(), mo_message::message(), and mo_constants::yeardays.

Referenced by read_forcing_nc().

Here is the call graph for this function:



Here is the caller graph for this function:



16.77.2.2 read_const_forcing_nc()

```
subroutine, public mo_read_forcing_nc::read_const_forcing_nc (
            character(len=*), intent(in) folder,
            integer(i4), intent(in) nRows,
            integer(i4), intent(in) nCols,
            character(len=*), intent(in) varName,
            logical, dimension(:,:), intent(in) mask,
            real(dp), dimension(:,:), intent(out), allocatable data )
```

Parameters

in	*folder*	

Author

Lennart Schueler, heavily influenced by read_forcing_nc

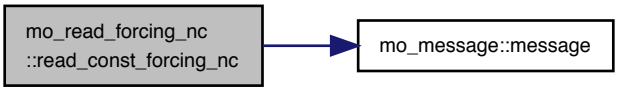
Date

May 2018

References mo_kind::dp, mo_kind::i4, and mo_message::message().

Referenced by mo_mrm_read_data::mrm_read_bankfull_runoff().

Here is the call graph for this function:



Here is the caller graph for this function:

16.77.2.3 read_forcing_nc()

```
subroutine, public mo_read_forcing_nc::read_forcing_nc (
             character(len = *), intent(in) folder,
             integer(i4), intent(in) nRows,
             integer(i4), intent(in) nCols,
             character(len = *), intent(in) varName,
             logical, dimension(:, :), intent(in) mask,
             real(dp), dimension(:, :, :), intent(out), allocatable data,
             type(period), intent(in), optional target_period,
             real(dp), intent(in), optional lower,
             real(dp), intent(in), optional upper,
             integer(i4), intent(in), optional nctimestep,
             character(256), intent(in), optional fileName,
             logical, intent(in), optional nocheck,
             logical, dimension(:, :, :), intent(out), optional, allocatable maskout )
```

Reads forcing input in NetCDF file format.

Reads netCDF forcing files. First, the dimensions given are cross-checked with header.txt information. Second, the data of the specified period are read from the specified directory. If the optional lower and/or upper bound for the

data values is given, the read data are checked for validity. The program is stopped if any value lies out of range. If the optinal argument nocheck is true, the data are not checked for coverage with the input mask. Additionally in this case an mask of vild data points can be received from the routine in maskout.

Parameters

in	*character(len = ∗) :: folder*	Name of the folder where data are stored
in	*integer(i4) :: nRows*	Number of datapoints in longitudinal direction
in	*integer(i4) :: nCols*	Number of datapoints in latitudinal direction
in	*character(len = ∗) :: varName*	Name of variable name to read
in	*logical, dimension(:, :) :: mask*	mask of valid data fields
out	*real(dp), dimension(:, :, :) :: data*	Data matrixdim_1 = longitude, dim_2 = latitude, dim_3 = time
in	*type(period), optional :: target_period*	Period the data are needed for
in	*real(dp), optional :: lower*	Lower bound for check of validity of data values
in	*real(dp), optional :: upper*	Upper bound for check of validity of data values
in	*integer(i4), optional :: nctimestep*	timestep in netcdf file
in	*character(256), optional :: fileName*	name of variable, defaults to fileName
in	*logical, optional :: nocheck*	.TRUE. if check for nodata values deactivateddefault = .FALSE. - check is done
out	*logical, dimension(:, :, :), optional :: maskout*	! mask of validdata points

Authors

Matthias Zink

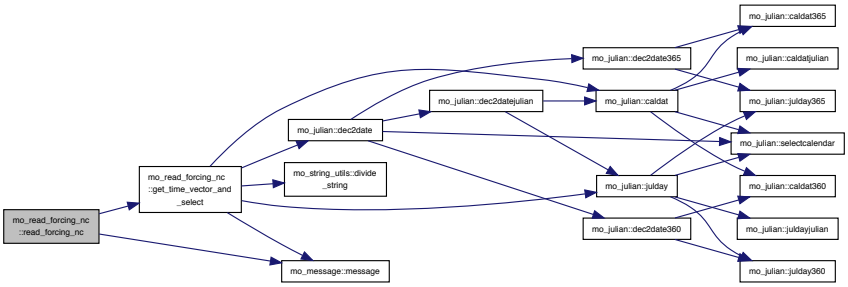
Date

May 2013

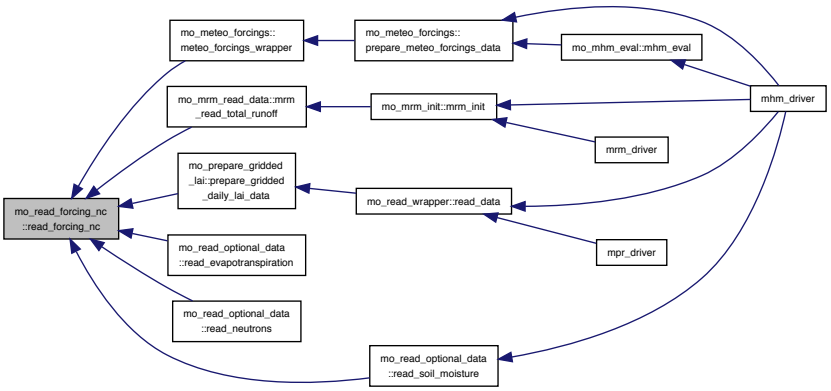
References mo_kind::dp, get_time_vector_and_select(), mo_kind::i4, and mo_message::message().

Referenced by mo_meteo_forcings::meteo_forcings_wrapper(), mo_mrm_read_data::mrm_read_total_runoff(), mo_prepare_gridded_lai::prepare_gridded_daily_lai_data(), mo_read_optional_data::read_evapotranspiration(), mo_read_optional_data::read_neutrons(), and mo_read_optional_data::read_soil_moisture().

Here is the call graph for this function:



Here is the caller graph for this function:



16.77.2.4 read_weights_nc()

```
subroutine, public mo_read_forcing_nc::read_weights_nc (
             character(len = *), intent(in) folder,
             integer(i4), intent(in) nRows,
             integer(i4), intent(in) nCols,
             character(len = *), intent(in) varName,
             real(dp), dimension(:, :, :, :), intent(out), allocatable data,
             logical, dimension(:, :), intent(in) mask,
             real(dp), intent(in), optional lower,
             real(dp), intent(in), optional upper,
             logical, intent(in), optional nocheck,
             logical, dimension(:, :, :, :), intent(out), optional, allocatable maskout,
             character(256), intent(in), optional fileName )
```

Reads weights for meteo forcings input in NetCDF file format.

Reads netCDF weight files. First, the dimensions given are cross-checked with header.txt information. If the optional lower and/or upper bound for the data values is given, the read data are checked for validity. The program is stopped if any value lies out of range. If the optinal argument nocheck is true, the data are not checked for coverage with the input mask. Additionally in this case an mask of vild data points can be received from the routine in maskout.

Parameters

in	*character(len = *) :: folder*	Name of the folder where data are stored
in	*integer(i4) :: nRows*	Number of datapoints in longitudinal direction
in	*integer(i4) :: nCols*	Number of datapoints in latitudinal direction
in	*character(len = *) :: varName*	Name of variable name to read
in	*logical, dimension(:, :) :: mask*	mask of valid data fields
out	*real(dp), dimension(:, :, :, :) :: data*	Data matrixdim_1 = longitude, dim_2 = latitude, dim_3 = months, dim_4 = hours
in	*real(dp), optional :: lower*	Lower bound for check of validity of data values
in	*real(dp), optional :: upper*	Upper bound for check of validity of data values
in	*logical, optional :: nocheck*	.TRUE. if check for nodata values deactivateddefault = .FALSE. - check is done

Parameters

in	*character(256), optional :: fileName*	name of variable, defaults to fileName
out	*logical, dimension(:, :, :), optional :: maskout*	! mask of validdata points

Authors

Stephan Thober & Matthias Zink

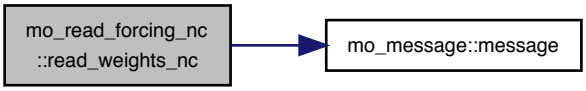
Date

Jan 2017

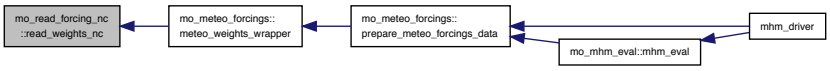
References mo_kind::dp, mo_kind::i4, and mo_message::message().

Referenced by mo_meteo_forcings::meteo_weights_wrapper().

Here is the call graph for this function:



Here is the caller graph for this function:



16.78 mo_read_latlon Module Reference

reading latitude and longitude coordinates for each basin

Functions/Subroutines

- subroutine, public read_latlon (ii, lon_var_name, lat_var_name, level_name, level)
  reads latitude and longitude coordinates

16.78.1 Detailed Description

reading latitude and longitude coordinates for each basin

TODO: add description

Authors

Stephan Thober

Date

Nov 2013

16.78.2 Function/Subroutine Documentation

16.78.2.1 read_latlon()

```
subroutine, public mo_read_latlon::read_latlon (
             integer(i4), intent(in) ii,
             character(*), intent(in) lon_var_name,
             character(*), intent(in) lat_var_name,
             character(*), intent(in) level_name,
             type(grid), intent(inout) level )
```

reads latitude and longitude coordinates

reads latitude and longitude coordinates from netcdf file for each basin and appends it to the global variables latitude and longitude.

Parameters

in	*integer(i4) :: ii*	basin indexFile name of the basins must be xxx_latlon.nc, wherexxx is the basin id. Variable names in the netcdf filehave to be 'lat' for latitude and 'lon' for longitude.
in	*character(∗) :: lon_var_name*	
in	*character(∗) :: lat_var_name*	
in	*character(∗) :: level_name*	
in,out	*type(Grid) :: level*	

Authors

Stephan Thober

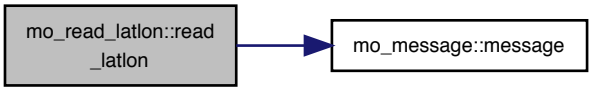
Date

Nov 2013

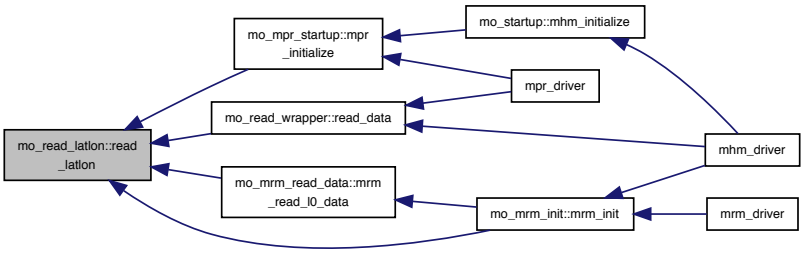
References mo_common_variables::filelatlon, and mo_message::message().

Referenced by mo_mpr_startup::mpr_initialize(), mo_mrm_init::mrm_init(), mo_mrm_read_data::mrm_read_l0_↩ data(), and mo_read_wrapper::read_data().

Here is the call graph for this function:



Here is the caller graph for this function:



16.79 mo_read_lut Module Reference

Routines reading lookup tables (lut).

Functions/Subroutines

- subroutine, public read_geoformation_lut (filename, fileunit, nGeo, geo_unit, geo_karstic)

  Reads LUT containing geological formation information.
- subroutine, public read_lai_lut (filename, fileunit, nLAI, LAIIDlist, LAI)

  Reads LUT containing LAI information.

16.79.1 Detailed Description

Routines reading lookup tables (lut).

This module contains routines reading various lookup tables (lut). (1) LUT containing gauge information. (2) LUT containing geological formation information. (3) LUT containing LAI class information.

Authors

Juliane Mai, Matthias Zink

Date

Jan 2013

16.79.2 Function/Subroutine Documentation

16.79.2.1 read_geoformation_lut()

```
subroutine, public mo_read_lut::read_geoformation_lut (
             character(len = *), intent(in) filename,
             integer(i4), intent(in) fileunit,
             integer(i4), intent(out) nGeo,
             integer(i4), dimension(:), intent(out), allocatable geo_unit,
             integer(i4), dimension(:), intent(out), allocatable geo_karstic )
```

Reads LUT containing geological formation information.

The LUT needs to have the following header:

```
nGeo_Formations  < Number of lines containing data >
GeoParam(i)   ClassUnit     Karstic      Description
```

The subsequent lines contains the geological formation information:

```
<GeoParam(i)>  <ClassUnit_i4>  <Karstic_i4>  <Description_char>
```

All following lines will be discarded while reading. GeoParam is a running index while ClassUnit is the unit of the map containing the geological formations such that it does not neccessarily contains subsequent numbers. The parametrization of this unit is part of the namelist mhm_parameter.nml under <geoparameter>.

Parameters

in	*character(len = *) :: filename*	File name of LUT
in	*integer(i4) :: fileunit*	Unit to open file
out	*integer(i4) :: nGeo*	Number of geological formations
out	*integer(i4), dimension(:) :: geo_unit*	List of id numbers of each geological formations
out	*integer(i4), dimension(:) :: geo_karstic*	ID of the Karstic formation (0 == does not exist)

Authors

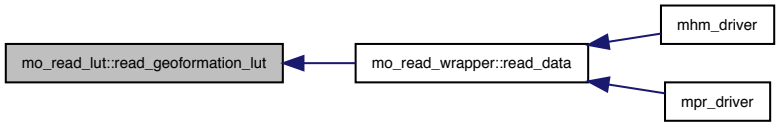
Juliane Mai

Date

Jan 2013

Referenced by mo_read_wrapper::read_data().

Here is the caller graph for this function:



16.79.2.2 read_lai_lut()

```
subroutine, public mo_read_lut::read_lai_lut (
            character(len = *), intent(in) filename,
            integer(i4), intent(in) fileunit,
            integer(i4), intent(out) nLAI,
            integer(i4), dimension(:), intent(out), allocatable LAIIDlist,
            real(dp), dimension(:, :), intent(out), allocatable LAI )
```

Reads LUT containing LAI information.

The LUT needs to have the following header:

```
NoLAIclasses  <Number of lines containing data>
Id  land-use  Jan.   Feb.    Mar.    Apr.    May    Jun.    Jul.    Aug.    Sep.    Oct.    Nov.    Dec
```

The subsequent lines contains the lai class information:

```
<ID_i4>  <landuse_char>  <val_1_dp>  <val_2_dp>  <val_3_dp>  <val_4_dp> ... <val_12_dp>
```

All following lines will be discarded while reading.

Parameters

in	*character(len = *) :: filename*	File name of LUT
in	*integer(i4) :: fileunit*	Unit to open file
out	*integer(i4) :: nLAI*	Number of LAI classes
out	*integer(i4), dimension(:) :: LAIIDlist*	List of ids of LAI classes
out	*real(dp), dimension(:, :) :: LAI*	LAI per class (row) and month (col)

Authors

Juliane Mai

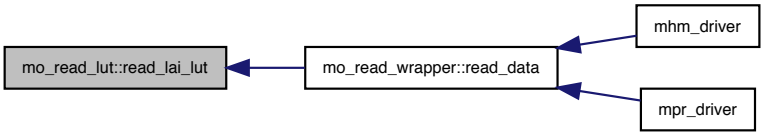
Date

Jan 2013

References mo_kind::i4, and mo_common_constants::yearmonths.

Referenced by mo_read_wrapper::read_data().

Here is the caller graph for this function:



16.80 mo_read_optional_data Module Reference

Read optional data for mHM calibration.

Functions/Subroutines

- subroutine, public read_soil_moisture (iBasin)

  Read soil moisture data from NetCDF file for calibration.
- subroutine, public read_basin_avg_tws

  Read basin average TWS timeseries from file, the same way runoff is read.
- subroutine, public read_neutrons (iBasin)

  Read neutrons data from NetCDF file for calibration.
- subroutine, public read_evapotranspiration (iBasin)

  Read evapotranspiration data from NetCDF file for calibration.

16.80.1 Detailed Description

Read optional data for mHM calibration.

Data have to be provided in resolution of the hydrology.

Authors

Matthias Zink

Date

Mar 2015

16.80.2 Function/Subroutine Documentation

16.80.2.1 read_basin_avg_tws()

```
subroutine, public mo_read_optional_data::read_basin_avg_tws ( )
```

Read basin average TWS timeseries from file, the same way runoff is read.

Read basin average TWS timeseries Allocate global basin_avg_TWS variable that contains the simulated values after the simulation.

Authors

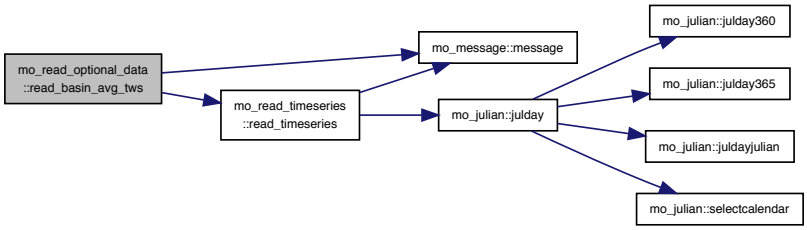
Oldrich Rakovec

Date

Oct 2015

References mo_global_variables::basin_avg_tws_obs, mo_global_variables::basin_avg_tws_sim, mo_common_mhm_mrm_variables::evalper, mo_message::message(), mo_common_variables::nbasins, mo_global_variables::nmeasperday_tws, mo_common_constants::nodata_dp, mo_common_mhm_mrm_variables::ntstepday, mo_common_mhm_mrm_variables::opti_function, mo_common_mhm_mrm_variables::optimize, mo_read_timeseries::read_timeseries(), mo_common_mhm_mrm_variables::simper, and mo_file::utws.

Here is the call graph for this function:



16.80.2.2 read_evapotranspiration()

```
subroutine, public mo_read_optional_data::read_evapotranspiration (
            integer(i4), intent(in) iBasin )
```

Read evapotranspiration data from NetCDF file for calibration.

This routine reads oberved evapotranspiration fields which are used for model calibration. The evapotranspiration file is expected to be called "et.nc" with a variable "et" inside. The data are read only for the evaluation period they are intended to be used for calibration. Evapotranspiration data are only read if one of the corresponding objective functions is chosen.

Parameters

in	*integer(i4) :: iBasin*	Basin Id

Authors

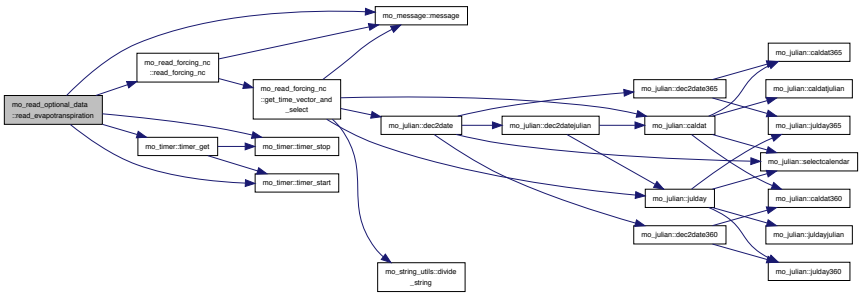
Johannes Brenner

Date

Feb 2017

References mo_global_variables::direvapotranspiration, mo_common_mhm_mrm_variables::evalper, mo_global_variables::l1_et, mo_global_variables::l1_et_mask, mo_common_variables::level1, mo_message::message(), mo_common_constants::nodata_dp, mo_global_variables::ntimesteps_l1_et, mo_read_forcing_nc::read_forcing_nc(), mo_timer::timer_get(), mo_timer::timer_start(), mo_timer::timer_stop(), and mo_global_variables::timestep_et_input.

Here is the call graph for this function:



16.80.2.3 read_neutrons()

```
subroutine, public mo_read_optional_data::read_neutrons (
            integer(i4), intent(in) iBasin )
```

Read neutrons data from NetCDF file for calibration.

This routine reads oberved neutron fields which are used for model calibration. The neutrons file is expected to be called "neutrons.nc" with a variable "neutrons" inside. The data are read only for the evaluation period they are intended to be used for calibration. Neutrons data are only read if one of the corresponding objective functions is chosen.

Parameters

in	*integer(i4) :: iBasin*	Basin Id

Authors

Martin Schroen

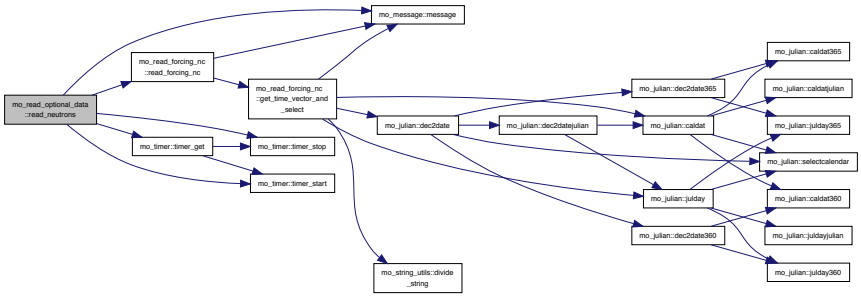
Date

Jul 2015

References mo_global_variables::dirneutrons, mo_common_mhm_mrm_variables::evalper, mo_global_variables::l1_neutronsdata, mo_global_variables::l1_neutronsdata_mask, mo_common_variables::level1, mo_message::message(), mo_common_constants::nodata_dp, mo_global_variables::ntimesteps_l1_neutrons, mo_read_

forcing_nc::read_forcing_nc(), mo_timer::timer_get(), mo_timer::timer_start(), mo_timer::timer_stop(), and mo_↩ global_variables::timestep_neutrons_input.

Here is the call graph for this function:



16.80.2.4 read_soil_moisture()

```
subroutine, public mo_read_optional_data::read_soil_moisture (
             integer(i4), intent(in) iBasin )
```

Read soil moisture data from NetCDF file for calibration.

This routine reads oberved soil moisture fields which are used for model calibration. The soil moisture file is expected to be called "sm.nc" with a variable "sm" inside. The data are read only for the evaluation period they are intended to be used for calibration. Soil moisture data are only read if one of the corresponding objective functions is chosen.

Parameters

in	*integer(i4) :: iBasin*	Basin Id

Authors

Matthias Zink

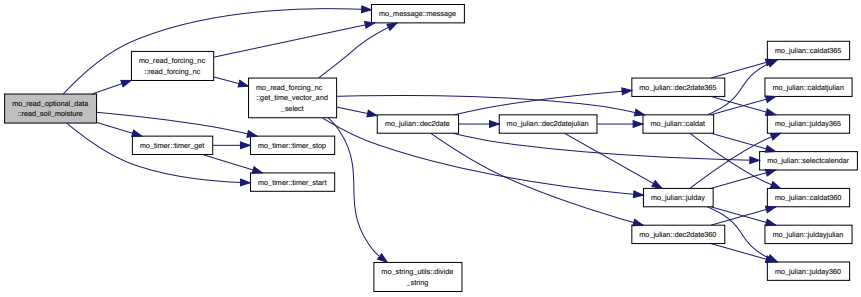
Date

Mar 2015

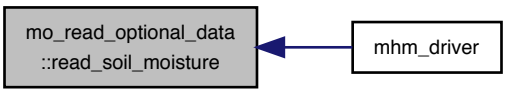
References mo_global_variables::dirsoil_moisture, mo_common_mhm_mrm_variables::evalper, mo_global_↩ variables::l1_sm, mo_global_variables::l1_sm_mask, mo_common_variables::level1, mo_message::message(), mo_common_constants::nodata_dp, mo_global_variables::ntimesteps_l1_sm, mo_read_forcing_nc::read_↩ forcing_nc(), mo_timer::timer_get(), mo_timer::timer_start(), mo_timer::timer_stop(), and mo_global_variables↩ ::timestep_sm_input.

Referenced by mhm_driver().

Here is the call graph for this function:



Here is the caller graph for this function:



16.81 mo_read_spatial_data Module Reference

Reads spatial input data.

Data Types

- interface read_spatial_data_ascii

  Reads spatial data files of ASCII format.

Functions/Subroutines

- subroutine read_spatial_data_ascii_dp (filename, fileunit, header_ncols, header_nrows, header_xllcorner, header_yllcorner, header_cellsize, data, mask)

  TODO: add description.

- subroutine read_spatial_data_ascii_i4 (filename, fileunit, header_ncols, header_nrows, header_xllcorner, header_yllcorner, header_cellsize, data, mask)

  TODO: add description.
- subroutine, public read_header_ascii (filename, fileunit, header_ncols, header_nrows, header_xllcorner, header_yllcorner, header_cellsize, header_nodata)

  Reads header lines of ASCII files.

16.81.1 Detailed Description

Reads spatial input data.

This module is to read spatial input data, e.g. dem, aspect, flow direction. The module provides a subroutine for ASCII files. (Subroutine for NetCDF files will come with release 5.1). The data are read from the specified directory.

Authors

Juliane Mai

Date

Dec 2012

16.81.2 Function/Subroutine Documentation

16.81.2.1 read_header_ascii()

```
subroutine, public mo_read_spatial_data::read_header_ascii (
             character(len = *), intent(in) filename,
             integer(i4), intent(in) fileunit,
             integer(i4), intent(out) header_ncols,
             integer(i4), intent(out) header_nrows,
             real(dp), intent(out) header_xllcorner,
             real(dp), intent(out) header_yllcorner,
             real(dp), intent(out) header_cellsize,
             real(dp), intent(out) header_nodata )
```

Reads header lines of ASCII files.

Reads header lines of ASCII files, e.g. dem, aspect, flow direction.

Parameters

in	*character(len = *) :: filename*	Name of file and its location
in	*integer(i4) :: fileunit*	File unit for open file
out	*integer(i4) :: header_nCols*	Reference number of columns
out	*integer(i4) :: header_nRows*	Reference number of rows
out	*real(dp) :: header_xllcorner*	Reference lower left corner (x)
out	*real(dp) :: header_yllcorner*	Reference lower left corner (y)
out	*real(dp) :: header_cellsize*	Reference cell size [m]
out	*real(dp) :: header_nodata*	Reference nodata value

Authors

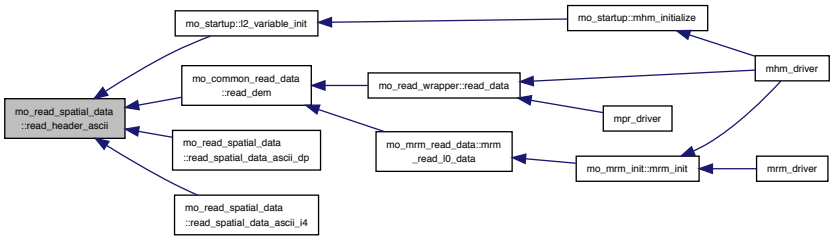
Juliane Mai

Date

Jan 2013

Referenced by mo_startup::l2_variable_init(), mo_common_read_data::read_dem(), read_spatial_data_ascii_dp(), and read_spatial_data_ascii_i4().

Here is the caller graph for this function:



16.81.2.2 read_spatial_data_ascii_dp()

```
subroutine mo_read_spatial_data::read_spatial_data_ascii_dp (
            character(len = *), intent(in) filename,
            integer(i4), intent(in) fileunit,
            integer(i4), intent(in) header_ncols,
            integer(i4), intent(in) header_nrows,
            real(dp), intent(in) header_xllcorner,
            real(dp), intent(in) header_yllcorner,
            real(dp), intent(in) header_cellsize,
            real(dp), dimension(:, :), intent(out), allocatable data,
            logical, dimension(:, :), intent(out), allocatable mask )  [private]
```

TODO: add description.

TODO: add description

Parameters

in	*character(len = *) :: filename*	filename with location
in	*integer(i4) :: fileunit*	unit for opening the file
in	*integer(i4) :: header_nCols*	number of columns of data fields:
in	*integer(i4) :: header_nRows*	number of rows of data fields:
in	*real(dp) :: header_xllcorner*	header read in lower left corner
in	*real(dp) :: header_yllcorner*	header read in lower left corner
in	*real(dp) :: header_cellsize*	header read in cellsize
out	*real(dp), dimension(:, :) :: data*	data
out	*logical, dimension(:, :) :: mask*	mask

Authors

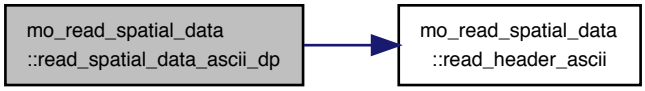
Robert Schweppe

Date

Jun 2018

References read_header_ascii().

Here is the call graph for this function:



16.81.2.3 read_spatial_data_ascii_i4()

```
subroutine mo_read_spatial_data::read_spatial_data_ascii_i4 (
             character(len = *), intent(in) filename,
             integer(i4), intent(in) fileunit,
             integer(i4), intent(in) header_ncols,
             integer(i4), intent(in) header_nrows,
             real(dp), intent(in) header_xllcorner,
             real(dp), intent(in) header_yllcorner,
             real(dp), intent(in) header_cellsize,
             integer(i4), dimension(:, :), intent(out), allocatable data,
             logical, dimension(:, :), intent(out), allocatable mask )  [private]
```

TODO: add description.

TODO: add description

Parameters

in	*character(len = *) :: filename*	filename with location
in	*integer(i4) :: fileunit*	unit for opening the file
in	*integer(i4) :: header_nCols*	number of columns of data fields:
in	*integer(i4) :: header_nRows*	number of rows of data fields:
in	*real(dp) :: header_xllcorner*	header read in lower left corner
in	*real(dp) :: header_yllcorner*	header read in lower left corner
in	*real(dp) :: header_cellsize*	header read in cellsize
out	*integer(i4), dimension(:, :) :: data*	data
out	*logical, dimension(:, :) :: mask*	mask

Authors

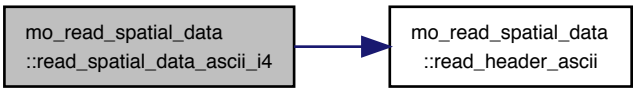
Robert Schweppe

Date

Jun 2018

References mo_kind::i4, and read_header_ascii().

Here is the call graph for this function:



16.82 mo_read_timeseries Module Reference

Routines to read files containing timeseries data.

Functions/Subroutines

- subroutine, public read_timeseries (filename, fileunit, periodStart, periodEnd, optimize, opti_function, data, mask, nMeasPerDay)

  Reads time series in ASCII format.

16.82.1 Detailed Description

Routines to read files containing timeseries data.

This routine is reading time series input data for a particular time period. The files need to have a specific header specified in the different routines.

Authors

Matthias Zink, Juliane Mai

Date

Jan 2013

16.82.2 Function/Subroutine Documentation

16.82.2.1 read_timeseries()

```
subroutine, public mo_read_timeseries::read_timeseries (
             character(len = *), intent(in) filename,
             integer(i4), intent(in) fileunit,
             integer(i4), dimension(3), intent(in) periodStart,
             integer(i4), dimension(3), intent(in) periodEnd,
             logical, intent(in) optimize,
             integer(i4), intent(in) opti_function,
             real(dp), dimension(:), intent(out), allocatable data,
             logical, dimension(:), intent(out), optional, allocatable mask,
             integer(i4), intent(out), optional nMeasPerDay )
```

Reads time series in ASCII format.

Reads time series in ASCII format. Needs specific header lines:

```
<description>
nodata  <nodata value>
n  <number of measurements per day>  measurements per day [1, 1440]
start  <YYYY_i4> <MM_i4> <DD_i4> <HH_i4> <MM_i4> (YYYY MM DD HH MM)
end    <YYYY_i4> <MM_i4> <DD_i4> <HH_i4> <MM_i4> (YYYY MM DD HH MM)
```

Line 6 is the first line with data in the following format:

```
<YYYY_i4> <MM_i4> <DD_i4> <HH_i4> <MM_i4> <data_dp>
```

The routine checks for missing data points and if data points are equal distanced. The first data point at each day has to be at HH:MM = 00:00.

Parameters

in	*character(len = *) :: filename*	File name
in	*integer(i4) :: fileunit*	Unit to open file
in	*integer(i4), dimension(3) :: periodStart*	Start day of reading (YYYY,MM,DD)
in	*integer(i4), dimension(3) :: periodEnd*	End day of reading (YYYY,MM,DD)
in	*logical :: optimize*	optimization flag
in	*integer(i4) :: opti_function*	
out	*real(dp), dimension(:) :: data*	Data vector
out	*logical, dimension(:), optional :: mask*	Mask for nodata values in data
out	*integer(i4), optional :: nMeasPerDay*	Number of data points per day

Authors

Matthias Zink, Juliane Mai

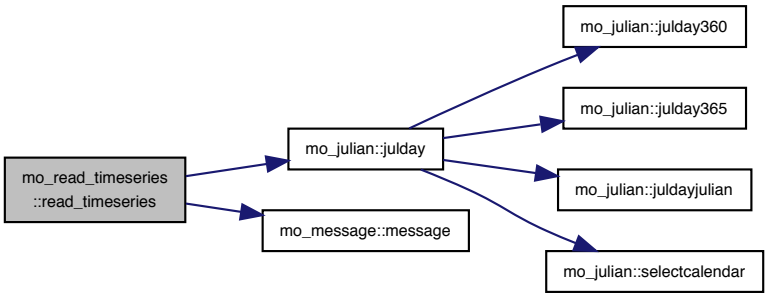
Date

Jan 2013

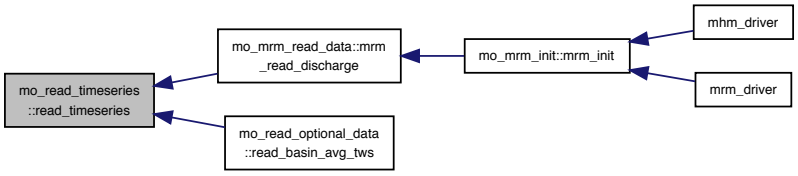
References mo_julian::julday(), and mo_message::message().

Referenced by mo_mrm_read_data::mrm_read_discharge(), and mo_read_optional_data::read_basin_avg_tws().

Here is the call graph for this function:



Here is the caller graph for this function:



16.83 mo_read_wrapper Module Reference

Wrapper for all reading routines.

Functions/Subroutines

- subroutine, public read_data (LAIPer)

  Reads data.
- subroutine check_consistency_lut_map (data, lookuptable, filename, unique_values)

  Checks if classes in input maps appear in look up tables.

16.83.1 Detailed Description

Wrapper for all reading routines.

This module is to wrap up all reading routines. The general written reading routines are used to store now the read data into global variables.

Authors

Juliane Mai, Matthias Zink

Date

Jan 2013

16.83.2 Function/Subroutine Documentation

16.83.2.1 check_consistency_lut_map()

```
subroutine mo_read_wrapper::check_consistency_lut_map (
            integer(i4), dimension(:), intent(in) data,
            integer(i4), dimension(:), intent(in) lookuptable,
            character(*), intent(in) filename,
            integer(i4), dimension(:), intent(out), optional, allocatable unique_values )
```

Checks if classes in input maps appear in look up tables.

Determines wether a class appearing in the morphological input is occuring in the respective look up table. mHM breaks if inconsistencies are discovered.

Parameters

in	*integer(i4), dimension(:) :: data*	map of study domain
in	*integer(i4), dimension(:) :: lookuptable*	look up table corresponding to map
in	*character(∗) :: filename*	name of the lut file - ERORR warn
out	*integer(i4), dimension(:), optional :: unique_values*	array of unique values in dataone

Authors

Matthias Zink

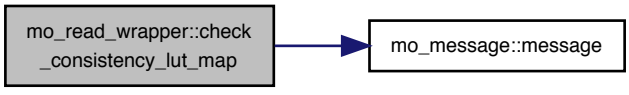
Date

Nov 2016

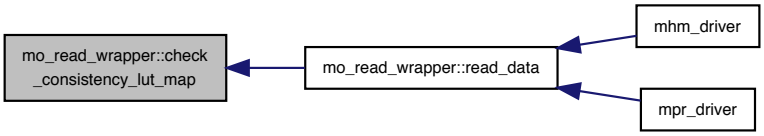
References mo_message::message().

Referenced by read_data().

Here is the call graph for this function:



Here is the caller graph for this function:



16.83.2.2 read_data()

```
subroutine, public mo_read_wrapper::read_data (
             type(period), dimension(:), intent(in), optional LAIPer )
```

Reads data.

The namelists are already read by read_config call. All LUTs are read from their respective directory and information within those files are shared across all basins to be modeled.

Parameters

in	*type(period), dimension(:), optional :: LAIPer*	

Authors

Juliane Mai & Matthias Zink

Date

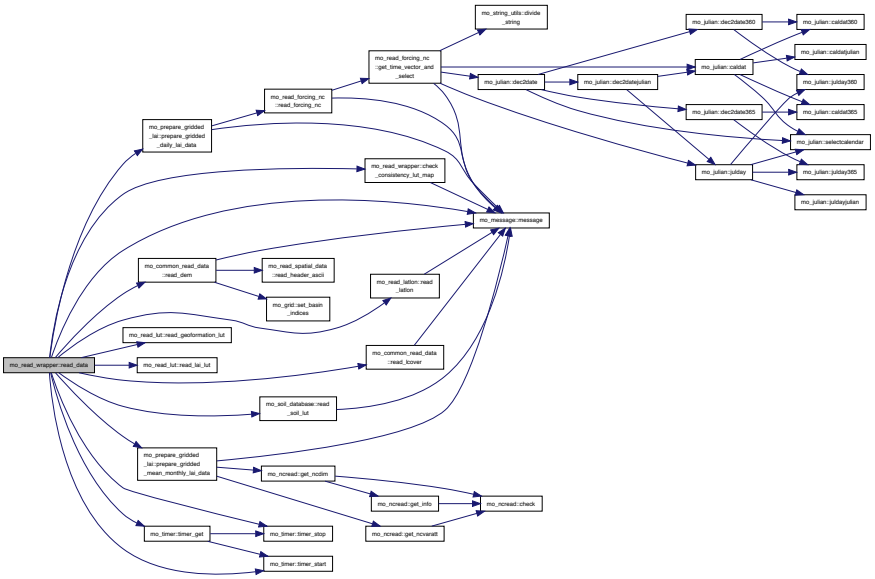
Feb 2013

by default; when iFlag_soilDB = 0

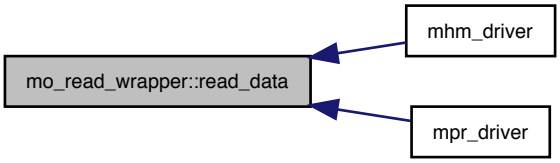
References check_consistency_lut_map(), mo_common_variables::dircommonfiles, mo_common_variables::dirmorpho, mo_mpr_file::file_aspect, mo_mpr_file::file_geolut, mo_mpr_file::file_hydrogeoclass, mo_mpr_file::file_laiclass, mo_mpr_file::file_lailut, mo_mpr_file::file_slope, mo_mpr_file::file_soil_database, mo_mpr_file::file_soil_database_1, mo_mpr_file::file_soilclass, mo_mpr_global_variables::geounitkar, mo_mpr_global_variables::geounitlist, mo_common_variables::global_parameters, mo_mpr_global_variables::iflag_soildb, mo_mpr_global_variables::l0_asp, mo_common_variables::l0_basin, mo_mpr_global_variables::l0_geounit, mo_mpr_global_variables::l0_gridded_lai, mo_mpr_global_variables::l0_slope, mo_mpr_global_variables::l0_soilid, mo_mpr_global_variables::lailut, mo_mpr_global_variables::laiunitlist, mo_common_variables::level0, mo_message::message(), mo_common_variables::nbasins, mo_mpr_global_variables::ngeounits, mo_mpr_global_variables::nlai, mo_mpr_global_variables::nlaiclass, mo_common_constants::nodata_dp, mo_common_constants::nodata_i4, mo_mpr_global_variables::nsoilhorizons_mhm, mo_prepare_gridded_lai::prepare_gridded_daily_lai_data(), mo_prepare_gridded_lai::prepare_gridded_mean_monthly_lai_data(), mo_common_variables::processmatrix, mo_common_read_data::read_dem(), mo_read_lut::read_geoformation_lut(), mo_read_lut::read_lai_lut(), mo_read_latlon::read_latlon(), mo_common_read_data::read_lcover(), mo_soil_database::read_soil_lut(), mo_mpr_global_variables::soildb, mo_timer::timer_get(), mo_timer::timer_start(), mo_timer::timer_stop(), mo_mpr_global_variables::timestep_lai_input, mo_mpr_file::uaspect, mo_mpr_file::ugeolut, mo_mpr_file::uhydrogeoclass, mo_mpr_file::ulaiclass, mo_mpr_file::ulailut, mo_mpr_file::uslope, mo_mpr_file::usoilclass, and mo_common_constants::yearmonths_i4.

Referenced by mhm_driver(), and mpr_driver().

Here is the call graph for this function:



Here is the caller graph for this function:



16.84 mo_restart Module Reference

reading and writing states, fluxes and configuration for restart of mHM.

Data Types

- interface unpack_field_and_write

  TODO: add description.

Functions/Subroutines

- subroutine, public write_restart_files (OutPath)

  write restart files for each basin

- subroutine, public read_restart_states (iBasin, InPath)

  reads fluxes and state variables from file
- subroutine unpack_field_and_write_1d_i4 (nc, var_name, var_dims, fill_value, data, mask, var_long_name)
- subroutine unpack_field_and_write_1d_dp (nc, var_name, var_dims, fill_value, data, mask, var_long_name)
- subroutine unpack_field_and_write_2d_dp (nc, var_name, var_dims, fill_value, data, mask, var_long_name)
- subroutine unpack_field_and_write_3d_dp (nc, var_name, var_dims, fill_value, data, mask, var_long_name)

16.84.1 Detailed Description

reading and writing states, fluxes and configuration for restart of mHM.

routines are seperated for reading and writing variables for:

- states and fluxes, and
- configuration. Reading of L11 configuration is also seperated from the rest, since it is only required when routing is activated.

Authors

Stephan Thober

Date

Jul 2013

16.84.2 Function/Subroutine Documentation

16.84.2.1 read_restart_states()

```
subroutine, public mo_restart::read_restart_states (
             integer(i4), intent(in) iBasin,
             character(256), intent(in) InPath )
```

reads fluxes and state variables from file

read fluxes and state variables from given restart directory and initialises all state variables that are initialized in the subroutine initialise, contained in module mo_startup.

Parameters

in	*integer(i4) :: iBasin*	number of basin
in	*character(256) :: InPath*	Input Path including trailing slash

Authors

Stephan Thober

Date

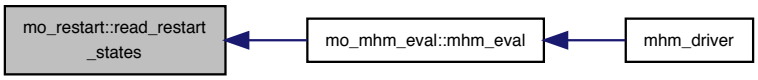
Apr 2013

References mo_kind::dp, mo_kind::i4, mo_mpr_global_variables::l1_aeroresist, mo_global_variables::l1_↩
aetcanopy, mo_global_variables::l1_aetsealed, mo_global_variables::l1_aetsoil, mo_mpr_global_variables::l1↩
_alpha, mo_global_variables::l1_baseflow, mo_mpr_global_variables::l1_degday, mo_mpr_global_variables::l1↩

_degdayinc, mo_mpr_global_variables::l1_degdaymax, mo_mpr_global_variables::l1_degdaynopre, mo_mpr_global_variables::l1_fasp, mo_global_variables::l1_fastrunoff, mo_mpr_global_variables::l1_froots, mo_mpr_global_variables::l1_fsealed, mo_mpr_global_variables::l1_harsamcoeff, mo_global_variables::l1_infilsoil, mo_global_variables::l1_inter, mo_mpr_global_variables::l1_jarvis_thresh_c1, mo_mpr_global_variables::l1_karstloss, mo_mpr_global_variables::l1_kbaseflow, mo_mpr_global_variables::l1_kfastflow, mo_mpr_global_variables::l1_kperco, mo_mpr_global_variables::l1_kslowflow, mo_mpr_global_variables::l1_maxinter, mo_global_variables::l1_melt, mo_global_variables::l1_percol, mo_mpr_global_variables::l1_petlaicorfactor, mo_global_variables::l1_preeffect, mo_mpr_global_variables::l1_prietayalpha, mo_global_variables::l1_rain, mo_global_variables::l1_runoffseal, mo_global_variables::l1_satstw, mo_mpr_global_variables::l1_sealedthresh, mo_global_variables::l1_sealstw, mo_global_variables::l1_slowrunoff, mo_global_variables::l1_snow, mo_global_variables::l1_snowpack, mo_global_variables::l1_soilmoist, mo_mpr_global_variables::l1_soilmoistexp, mo_mpr_global_variables::l1_soilmoistfc, mo_mpr_global_variables::l1_soilmoistsat, mo_mpr_global_variables::l1_surfresist, mo_mpr_global_variables::l1_tempthresh, mo_global_variables::l1_throughfall, mo_global_variables::l1_total_runoff, mo_global_variables::l1_unsatstw, mo_mpr_global_variables::l1_unsatthresh, mo_mpr_global_variables::l1_wiltingpoint, mo_common_variables::lc_year_end, mo_common_variables::lc_year_start, mo_common_variables::level1, mo_mpr_global_variables::nlai, mo_common_variables::nlcoverscene, mo_mpr_global_variables::nsoilhorizons_mhm, and mo_common_variables::processmatrix.

Referenced by mo_mhm_eval::mhm_eval().

Here is the caller graph for this function:



16.84.2.2 unpack_field_and_write_1d_dp()

```
subroutine mo_restart::unpack_field_and_write_1d_dp (
             type(ncdataset), intent(inout) nc,
             character(*), intent(in) var_name,
             type(ncdimension), dimension(:), intent(in) var_dims,
             real(dp), intent(in) fill_value,
             real(dp), dimension(:), intent(in) data,
             logical, dimension(:, :), intent(in) mask,
             character(*), intent(in), optional var_long_name )
```

References mo_kind::dp.

16.84.2.3 unpack_field_and_write_1d_i4()

```
subroutine mo_restart::unpack_field_and_write_1d_i4 (
             type(ncdataset), intent(inout) nc,
             character(*), intent(in) var_name,
             type(ncdimension), dimension(:), intent(in) var_dims,
             integer(i4), intent(in) fill_value,
             integer(i4), dimension(:), intent(in) data,
             logical, dimension(:, :), intent(in) mask,
             character(*), intent(in), optional var_long_name )
```

References mo_kind::i4.

16.84.2.4 unpack_field_and_write_2d_dp()

```
subroutine mo_restart::unpack_field_and_write_2d_dp (
             type(ncdataset), intent(inout) nc,
             character(*), intent(in) var_name,
             type(ncdimension), dimension(:), intent(in) var_dims,
             real(dp), intent(in) fill_value,
             real(dp), dimension(:, :), intent(in) data,
             logical, dimension(:, :), intent(in) mask,
             character(*), intent(in), optional var_long_name )
```

References mo_kind::dp, and mo_kind::i4.

16.84.2.5 unpack_field_and_write_3d_dp()

```
subroutine mo_restart::unpack_field_and_write_3d_dp (
             type(ncdataset), intent(inout) nc,
             character(*), intent(in) var_name,
             type(ncdimension), dimension(:), intent(in) var_dims,
             real(dp), intent(in) fill_value,
             real(dp), dimension(:, :, :), intent(in) data,
             logical, dimension(:, :), intent(in) mask,
             character(*), intent(in), optional var_long_name )
```

References mo_kind::dp, and mo_kind::i4.

16.84.2.6 write_restart_files()

```
subroutine, public mo_restart::write_restart_files (
             character(256), dimension(:), intent(in) OutPath )
```

write restart files for each basin

write restart files for each basin. For each basin three restart files are written. These are xxx_states.nc, xxx_L11↩ _config.nc, and xxx_config.nc (xxx being the three digit basin index). If a variable is added here, it should also be added in the read restart routines below.

Parameters

in	*character(256), dimension(:) :: OutPath*	Output Path for each basin

Authors

Stephan Thober

Date

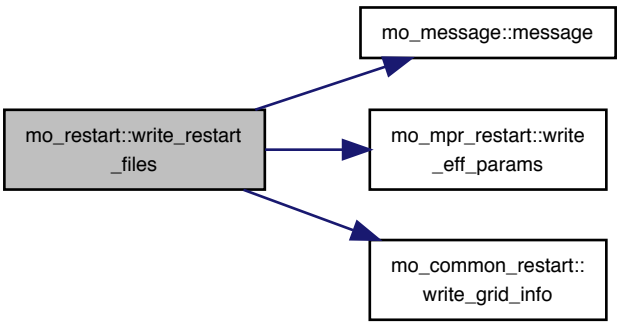
Jun 2014

References mo_kind::dp, mo_kind::i4, mo_global_variables::l1_aetcanopy, mo_global_variables::l1_aetsealed, mo_global_variables::l1_aetsoil, mo_global_variables::l1_baseflow, mo_global_variables::l1_fastrunoff, mo_↩

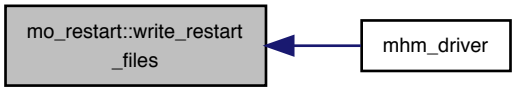
global_variables::l1_infilsoil, mo_global_variables::l1_inter, mo_global_variables::l1_melt, mo_global_variables::l1_percol, mo_global_variables::l1_preeffect, mo_global_variables::l1_rain, mo_global_variables::l1_runoffseal, mo_global_variables::l1_satstw, mo_global_variables::l1_sealstw, mo_global_variables::l1_slowrunoff, mo_global_variables::l1_snow, mo_global_variables::l1_snowpack, mo_global_variables::l1_soilmoist, mo_global_variables::l1_throughfall, mo_global_variables::l1_total_runoff, mo_global_variables::l1_unsatstw, mo_common_variables::level1, mo_message::message(), mo_mpr_global_variables::nlai, mo_common_variables::nlcoverscene, mo_common_constants::nodata_dp, mo_mpr_global_variables::nsoilhorizons_mhm, mo_mpr_restart::write_eff_params(), and mo_common_restart::write_grid_info().

Referenced by mhm_driver().

Here is the call graph for this function:



Here is the caller graph for this function:



16.85 mo_runoff Module Reference

Runoff generation for the unsaturated zone, saturated zone (or groundwater zone), and runoff accumulation.

Functions/Subroutines

- subroutine, public runoff_unsat_zone (k1, kp, k0, alpha, karst_loss, pefec_soil, unsat_thresh, sat_storage, unsat_storage, slow_interflow, fast_interflow, perc)

  Runoff generation for the saturated zone.
- subroutine, public runoff_sat_zone (k2, sat_storage, baseflow)

  Runoff generation for the saturated zone.

- subroutine, public l1_total_runoff (fSealed_area_fraction, fast_interflow, slow_interflow, baseflow, direct_↩ runoff, total_runoff)

  total runoff accumulation at level 1

16.85.1 Detailed Description

Runoff generation for the unsaturated zone, saturated zone (or groundwater zone), and runoff accumulation.

This module generates the runoff for the unsaturated and saturated zones and provides runoff accumulation.

Authors

Vladyslav Prykhodko

Date

Dec 2012

16.85.2 Function/Subroutine Documentation

16.85.2.1 l1_total_runoff()

```
subroutine, public mo_runoff::l1_total_runoff (
             real(dp), intent(in) fSealed_area_fraction,
             real(dp), intent(in) fast_interflow,
             real(dp), intent(in) slow_interflow,
             real(dp), intent(in) baseflow,
             real(dp), intent(in) direct_runoff,
             real(dp), intent(out) total_runoff )
```

total runoff accumulation at level 1

Accumulates runoff.

$$q_T = (q_0 + q_1 + q_2) * (1 - fSealed) + q_D * fSealed$$

, where fSealed is the fraction of sealed area.

Parameters

in	*REAL(dp) :: fSealed_area_fraction*	sealed area fraction [1]
in	*REAL(dp) :: fast_interflow*	q_0 Fast runoff component [mm tst-1]
in	*REAL(dp) :: slow_interflow*	q_1 Slow runoff component [mm tst-1]
in	*REAL(dp) :: baseflow*	q_2 Baseflow [mm tsts-1]
in	*REAL(dp) :: direct_runoff*	q_D Direct runoff from impervious areas [mm tst-1]
out	*REAL(dp) :: total_runoff*	q_T Generated runoff [mm tst-1]

Authors

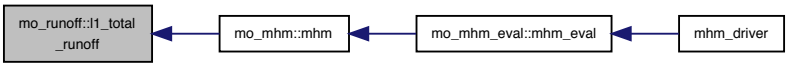
Vladyslav Prykhodko

Date

Dec 2012

Referenced by mo_mhm::mhm().

Here is the caller graph for this function:



16.85.2.2 runoff_sat_zone()

```
subroutine, public mo_runoff::runoff_sat_zone (
             real(dp), intent(in) k2,
             real(dp), intent(inout) sat_storage,
             real(dp), intent(out) baseflow )
```

Runoff generation for the saturated zone.

Calculates the runoff generation for the saturated zone. If the level of the ground water reservoir is zero, then the baseflow is also zero. If the level of the ground water reservoir is greater than zero, then the baseflow is equal to baseflow recession coefficient times the level of the ground water reservoir, which will be then reduced by the value of baseflow.

Parameters

in	*REAL(dp) :: k2*	Baseflow recession coefficient [d-1]
in,out	*REAL(dp) :: sat_storage*	Groundwater storage [mm]
out	*REAL(dp) :: baseflow*	Baseflow [mm d-1]

Authors

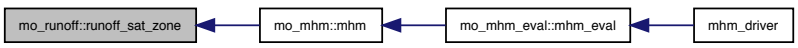
Vladyslav Prykhodko

Date

Dec 2012

Referenced by mo_mhm::mhm().

Here is the caller graph for this function:



16.85.2.3 runoff_unsat_zone()

```
subroutine, public mo_runoff::runoff_unsat_zone (
             real(dp), intent(in) k1,
             real(dp), intent(in) kp,
             real(dp), intent(in) k0,
             real(dp), intent(in) alpha,
             real(dp), intent(in) karst_loss,
             real(dp), intent(in) pefec_soil,
             real(dp), intent(in) unsat_thresh,
             real(dp), intent(inout) sat_storage,
             real(dp), intent(inout) unsat_storage,
             real(dp), intent(out) slow_interflow,
             real(dp), intent(out) fast_interflow,
             real(dp), intent(out) perc )
```

Runoff generation for the saturated zone.

Calculates the runoff generation for the unsaturated zone. Calculates percolation, interflow and baseflow. Updates upper soil and groundwater storages.

Parameters

in	*REAL(dp) :: k1*	Recession coefficient of the upper reservoir,lower outlet [d-1]
in	*REAL(dp) :: kp*	Percolation coefficient [d-1]
in	*REAL(dp) :: k0*	Recession coefficient of the upperreservoir, upper outlet [d-1]
in	*REAL(dp) :: alpha*	Exponent for the upper reservoir [-]
in	*REAL(dp) :: karst_loss*	Karstic percolation loss [-]
in	*REAL(dp) :: pefec_soil*	Input to the soil layer [mm]
in	*REAL(dp) :: unsat_thresh*	Threshold water depth in upper reservoir(for Runoff contribution) [mm]
in,out	*REAL(dp) :: sat_storage*	Groundwater storage [mm]
in,out	*REAL(dp) :: unsat_storage*	Upper soil storage [mm]
out	*REAL(dp) :: slow_interflow*	Slow runoff component [mm d-1]
out	*REAL(dp) :: fast_interflow*	Fast runoff component [mm d-1]
out	*REAL(dp) :: perc*	Percolation [mm d-1]

Authors

Vladyslav Prykhodko

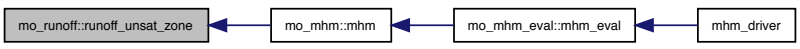
Date

Dec 2012

References mo_common_constants::eps_dp.

Referenced by mo_mhm::mhm().

Here is the caller graph for this function:



16.86 mo_sce Module Reference

Shuffled Complex Evolution optimization algorithm.

Functions/Subroutines

- real(dp) function, dimension(size(pini, 1)), public sce (eval, functn, pini, prange, mymaxn, mymaxit, mykstop, mypcento, mypeps, myseed, myngs, mynpg, mynps, mynspl, mymings, myiniflg, myprint, mymask, myalpha, mybeta, tmp_file, popul_file, popul_file_append, parallel, restart, restart_file, bestf, neval, history)

  Shuffled Complex Evolution (SCE) algorithm for global optimization.
- subroutine parstt (x, bound, peps, mask, xnstd, gnrng, ipcnvg)
- subroutine comp (ngs2, npg, a, af, b, bf)
- subroutine sort_matrix (rb, ra)
- subroutine chkcst (x, bl, bu, mask, ibound)
- subroutine getpnt (idist, bl, bu, std, xi, mask, save_state, x)
- subroutine cce (s, sf, bl, bu, maskpara, xnstd, icall, maxn, maxit, save_state_gauss, functn, eval, alpha, beta, history, idot)

16.86.1 Detailed Description

Shuffled Complex Evolution optimization algorithm.

Optimization algorithm using Shuffled Complex Evolution strategy. Original version 2.1 of Qingyun Duan (1992) rewritten in Fortran 90.

Authors

Juliane Mai

Date

Feb 2013

16.86.2 Function/Subroutine Documentation

16.86.2.1 cce()

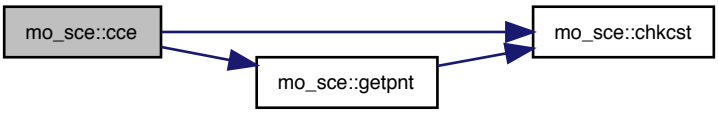
```
subroutine mo_sce::cce (
             real(dp), dimension(:, :), intent(inout) s,
             real(dp), dimension(:), intent(inout) sf,
             real(dp), dimension(:), intent(in) bl,
             real(dp), dimension(:), intent(in) bu,
             logical, dimension(:), intent(in) maskpara,
             real(dp), dimension(:), intent(in) xnstd,
             integer(i8), intent(inout) icall,
             integer(i8), intent(in) maxn,
             logical, intent(in) maxit,
             integer(i8), dimension(n_save_state), intent(inout) save_state_gauss,
             procedure(objective_interface), intent(in), pointer functn,
             procedure(eval_interface), intent(in), pointer eval,
             real(dp), intent(in) alpha,
             real(dp), intent(in) beta,
```

```
            real(dp), dimension(maxn), intent(inout) history,
            logical, intent(in) idot )
```

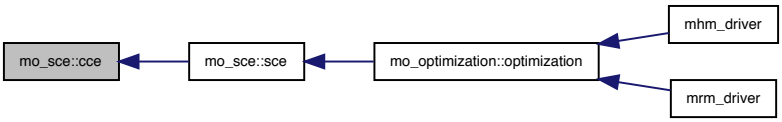
References chkcst(), mo_kind::dp, getpnt(), mo_kind::i4, mo_kind::i8, and mo_xor4096::n_save_state.

Referenced by sce().

Here is the call graph for this function:



Here is the caller graph for this function:



16.86.2.2 chkcst()

```
subroutine mo_sce::chkcst (
            real(dp), dimension(:), intent(in) x,
            real(dp), dimension(:), intent(in) bl,
            real(dp), dimension(:), intent(in) bu,
            logical, dimension(:), intent(in) mask,
            integer(i4), intent(out) ibound )
```

References mo_kind::dp, and mo_kind::i4.

Referenced by cce(), and getpnt().

Here is the caller graph for this function:

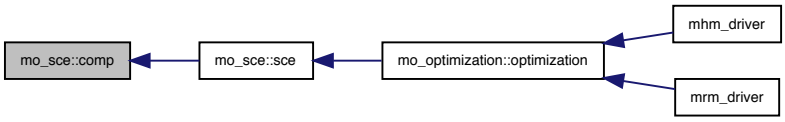
16.86.2.3 comp()

```
subroutine mo_sce::comp (
             integer(i4), intent(in) ngs2,
             integer(i4), intent(in) npg,
             real(dp), dimension(:, :), intent(inout) a,
             real(dp), dimension(:), intent(inout) af,
             real(dp), dimension(size(a, 1), size(a, 2)), intent(out) b,
             real(dp), dimension(size(af)), intent(out) bf )
```

References mo_kind::dp, and mo_kind::i4.

Referenced by sce().

Here is the caller graph for this function:



16.86.2.4 getpnt()

```
subroutine mo_sce::getpnt (
             integer(i4), intent(in) idist,
             real(dp), dimension(:), intent(in) bl,
             real(dp), dimension(:), intent(in) bu,
             real(dp), dimension(:), intent(in) std,
             real(dp), dimension(:), intent(in) xi,
             logical, dimension(:), intent(in) mask,
             integer(i8), dimension(n_save_state), intent(inout) save_state,
             real(dp), dimension(size(xi, 1)), intent(out) x )
```

References chkcst(), mo_kind::dp, mo_kind::i4, mo_kind::i8, and mo_xor4096::n_save_state.

Referenced by cce(), and sce().

Here is the call graph for this function:

Here is the caller graph for this function:

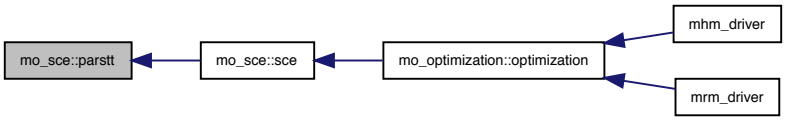
16.86.2.5 parstt()

```
subroutine mo_sce::parstt (
            real(dp), dimension(:, :), intent(in) x,
            real(dp), dimension(:), intent(in) bound,
            real(dp), intent(in) peps,
            logical, dimension(:), intent(in) mask,
            real(dp), dimension(size(bound, 1)), intent(out) xnstd,
            real(dp), intent(out) gnrng,
            integer(i4), intent(out) ipcnvg )  [private]
```

References mo_kind::dp, and mo_kind::i4.

Referenced by sce().

Here is the caller graph for this function:



16.86.2.6 sce()

```
real(dp) function, dimension(size(pini, 1)), public mo_sce::sce (
            procedure(eval_interface), intent(in), pointer eval,
            procedure(objective_interface), intent(in), pointer functn,
            real(dp), dimension(:), intent(in) pini,
            real(dp), dimension(:, :), intent(in) prange,
            integer(i8), intent(in), optional mymaxn,
            logical, intent(in), optional mymaxit,
            integer(i4), intent(in), optional mykstop,
            real(dp), intent(in), optional mypcento,
            real(dp), intent(in), optional mypeps,
            integer(i8), intent(in), optional myseed,
            integer(i4), intent(in), optional myngs,
            integer(i4), intent(in), optional mynpg,
```

```
integer(i4), intent(in), optional mynps,
integer(i4), intent(in), optional mynspl,
integer(i4), intent(in), optional mymings,
integer(i4), intent(in), optional myiniflg,
integer(i4), intent(in), optional myprint,
logical, dimension(size(pini, 1)), intent(in), optional mymask,
real(dp), intent(in), optional myalpha,
real(dp), intent(in), optional mybeta,
character(len = *), intent(in), optional tmp_file,
character(len = *), intent(in), optional popul_file,
logical, intent(in), optional popul_file_append,
logical, intent(in), optional parallel,
logical, intent(in), optional restart,
character(len = *), intent(in), optional restart_file,
real(dp), intent(out), optional bestf,
integer(i8), intent(out), optional neval,
real(dp), dimension(:), intent(out), optional, allocatable history )
```

Shuffled Complex Evolution (SCE) algorithm for global optimization.

Shuffled Complex Evolution method for global optimization
– version 2.1

by Qingyun Duan
Department of Hydrology & Water Resources
University of Arizona, Tucson, AZ 85721
(602) 621-9360, email: duan@hwr.arizona.edu

Written by Qingyun Duan, Oct 1990.
Revised by Qingyun Duan, Aug 1991.
Revised by Qingyun Duan, Apr 1992.

Re-written by Juliane Mai, Feb 2013.

Statement by Qingyun Duan:

This general purpose global optimization program is developed at the Department of Hydrology & Water Resources of the University of Arizona. Further information regarding the SCE-UA method can be obtained from Dr. Q. Duan, Dr. S. Sorooshian or Dr. V.K. Gupta at the address and phone number listed above. We request all users of this program make proper reference to the paper entitled 'Effective and Efficient Global Optimization for Conceptual Rainfall-runoff Models' by Duan, Q., S. Sorooshian, and V.K. Gupta, Water Resources Research, Vol 28(4), pp.1015-1031, 1992.

The function to be minimized is the first argument of DDS and must be defined as

```
function functn(p)
    use mo_kind, only: dp
    implicit none
    real(dp), dimension(:), intent(in) :: p
    real(dp) :: functn
end function functn
```

Parameters

in	*real(dp) :: functn(p)*	Function on which to search the optimum
in	*real(dp) :: pini(:)*	inital value of decision variables
in	*real(dp) :: prange(size(pini),2)*	Min/max range of decision variables
in	*integer(i8), optional :: mymaxn*	max no. of trials allowed before optimization is terminated DEFAULT: 1000_i8

Parameters

in	*logical, optional :: mymaxit*	maximization (.true.) or minimization (.false.) of function DEFAULT: false
in	*integer(i4), optional :: mykstop*	number of shuffling loops in which the criterion value must change by given percentage before optimiz. is terminated DEFAULT: 10_i4
in	*real(dp), optional :: mypcento*	percentage by which the criterion value must change in given number of shuffling loops DEFAULT: 0.0001_dp
in	*real(dp), optional :: mypeps*	optimization is terminated if volume of complex has converged to given percentage of feasible space DEFAULT: 0.001_dp
in	*integer(i8), optional :: myseed*	initial random seed DEFAULT: get_timeseed
in	*integer(i4), optional :: myngs*	number of complexes in the initial population DEFAULT: 2_i4
in	*integer(i4), optional :: mynpg*	number of points in each complex DEFAULT: 2∗n+1
in	*integer(i4), optional :: mynps*	number of points in a sub-complex DEFAULT: n+1
in	*integer(i4), optional :: mynspl*	number of evolution steps allowed for each complex before complex shuffling DEFAULT: 2∗n+1
in	*integer(i4), optional :: mymings*	minimum number of complexes required, if the number of complexes is allowed to reduce as the optimization proceeds DEFAULT: ngs = number of complexes in initial population
in	*integer(i4), optional :: myiniflg*	flag on whether to include the initial point in population 0, not included 1, included (DEFAULT)
in	*integer(i4), optional :: myprint*	flag for controlling print-out after each shuffling loop 0, print information on the best point of the population 1, print information on every point of the population 2, no printing (DEFAULT) 3, same as 0 but print progress '.' on every function call 4, same as 1 but print progress '.' on every function call
in	*logical, optional :: mymask(size(pini))*	parameter included in optimization (true) or discarded (false) DEFAULT: .true.
in	*real(dp), optional :: myalpha*	parameter for reflection of points in complex DEFAULT: 0.8_dp
in	*real(dp), optional :: mybeta*	parameter for contraction of points in complex DEFAULT: 0.45_dp
in	*character(len=∗), optional :: tmp_file*	if given: write results after each evolution loop to temporal output file of that name **of headlines: 7**

format: '# nloop icall ngs1 bestf worstf ...
... gnrng (bestx(j),j=1,nn)'

Parameters

in	*character(len=∗), optional :: popul_file*	if given: write whole population to file of that name **of headlines: 1**

format: #_evolution_loop, xf(i), (x(i,j),j=1,nn)
total number of lines written <= neval <= mymaxn

Parameters

in	*logical, optional :: popul_file_append*	if true, append to existing population file (default: false)
in	*logical, optional :: parallel*	sce runs in parallel (true) or not (false) parallel sce should only be used if model/ objective is not parallel DEFAULT: .false.
in	*logical, optional :: restart*	if .true.: restart former sce run from restart_file
in	*character(len=∗), optional :: restart_file*	file name for read/write of restart file (default: mo_sce.restart)
out	*real(dp), optional :: bestf*	the best value of the function.
out	*integer(i8), optional :: neval*	number of function evaluations needed.
out	*real(dp), optional, allocatable :: history(:)*	the history of best function values, history(neval)=bestf

Returns

real(dp) :: bestx(size(pini)) — The parameters of the point which is estimated to minimize/maximize the function.

Note

Maximal number of parameters is 1000.
SCE is OpenMP enabled on the loop over the complexes.
OMP_NUM_THREADS > 1 does not give reproducible results even when seeded!

Author

Juliane Mai

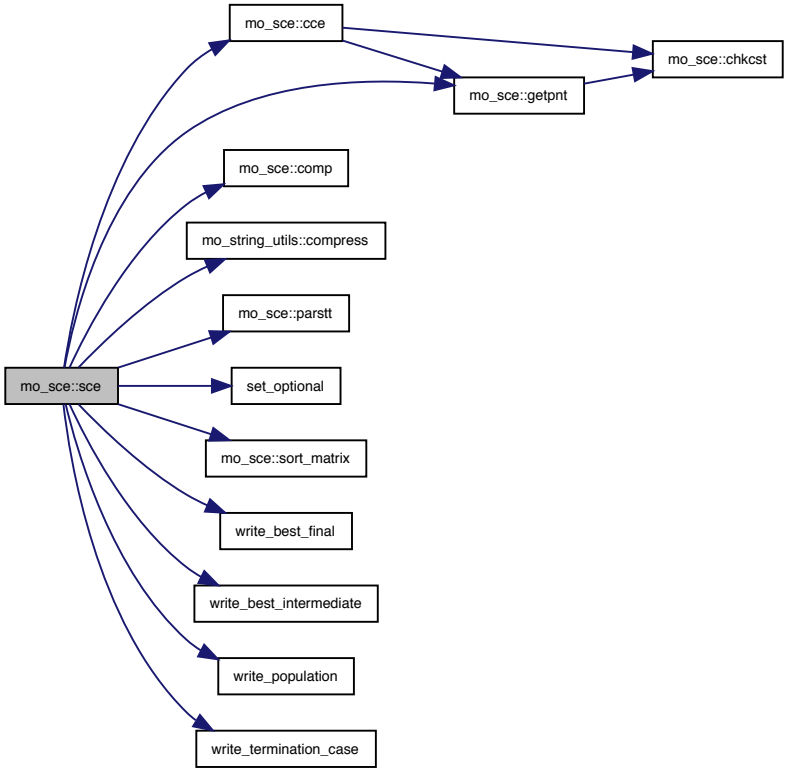
Date

Feb 2013

References cce(), comp(), mo_string_utils::compress(), mo_kind::dp, getpnt(), mo_kind::i4, mo_kind::i8, mo_↩xor4096::n_save_state, parstt(), set_optional(), sort_matrix(), write_best_final(), write_best_intermediate(), write↩_population(), and write_termination_case().

Referenced by mo_optimization::optimization().

Here is the call graph for this function:



Here is the caller graph for this function:

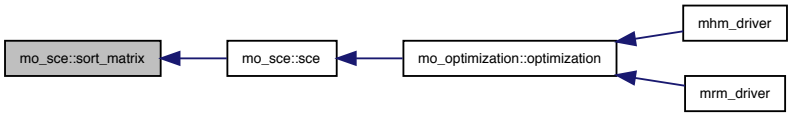
16.86.2.7 sort_matrix()

```
subroutine mo_sce::sort_matrix (
             real(dp), dimension(:, :), intent(inout) rb,
             real(dp), dimension(:), intent(inout) ra )
```

References mo_kind::dp, and mo_kind::i4.

Referenced by sce().

Here is the caller graph for this function:



16.87 mo_set_netcdf_outputs Module Reference

Defines the structure of the netCDF to write the output in.

Functions/Subroutines

- subroutine, public set_netcdf (NoNetcdfVars, nrows, ncols)

  Runs mhm with a specific parameter set and returns required variables, e.g. runoff.

16.87.1 Detailed Description

Defines the structure of the netCDF to write the output in.

All output variables are initialized for the NetCDF.

Authors

Matthias Zink

Date

Apr 2013

16.87.2 Function/Subroutine Documentation

16.87.2.1 set_netcdf()

```
subroutine, public mo_set_netcdf_outputs::set_netcdf (
            integer(i4), intent(in) NoNetcdfVars,
            integer(i4), intent(in) nrows,
            integer(i4), intent(in) ncols )
```

Runs mhm with a specific parameter set and returns required variables, e.g. runoff.

Runs mhm with a specific parameter set and returns required variables, e.g. runoff.

Parameters

in	*integer(i4) :: NoNetcdfVars*	
in	*integer(i4) :: nrows*	
in	*integer(i4) :: ncols*	

Authors

Matthias Zink

Date

Apr 2013

References mo_ncwrite::dnc, mo_ncwrite::ndims, mo_ncwrite::nvars, and mo_ncwrite::v.

16.88 mo_snow_accum_melt Module Reference

Snow melting and accumulation.

Functions/Subroutines

- subroutine, public snow_accum_melt (deg_day_incr, deg_day_max, deg_day_noprec, prec, temperature, temperature_thresh, thrfall, snow_pack, deg_day, melt, prec_effect, rain, snow)

  Snow melting and accumulation.

16.88.1 Detailed Description

Snow melting and accumulation.

This module calculates snow melting and accumulation.

Authors

Vladyslav Prykhodko

Date

Dec 2012

16.88.2 Function/Subroutine Documentation

16.88.2.1 snow_accum_melt()

```
subroutine, public mo_snow_accum_melt::snow_accum_melt (
             real(dp), intent(in) deg_day_incr,
             real(dp), intent(in) deg_day_max,
             real(dp), intent(in) deg_day_noprec,
             real(dp), intent(in) prec,
             real(dp), intent(in) temperature,
             real(dp), intent(in) temperature_thresh,
             real(dp), intent(in) thrfall,
             real(dp), intent(inout) snow_pack,
             real(dp), intent(out) deg_day,
             real(dp), intent(out) melt,
             real(dp), intent(out) prec_effect,
             real(dp), intent(out) rain,
             real(dp), intent(out) snow )
```

Snow melting and accumulation.

Separates throughfall into rain and snow by comparing the temperature with the treshhold. by comparing the temperature with the treshhold. Calculates degree daily factor. Calculates snow melting rates. Calculates snow, rain and effective precipitation depth and snow pack.

Parameters

in	*REAL(dp) :: deg_day_incr*	Increase of the Degree-day factor per mm of increasein precipitation [s-1 degreeC-1]
in	*REAL(dp) :: deg_day_max*	Maximum Degree-day factor [m-1 degreeC-1]
in	*REAL(dp) :: deg_day_noprec*	Degree-day factor with no precipitation [m-1 degreeC-1]
in	*REAL(dp) :: prec*	Daily mean precipitation [m]
in	*REAL(dp) :: temperature*	Daily mean temperature [degreeC]
in	*REAL(dp) :: temperature_thresh*	Threshold temperature for snow/rain [degreeC]
in	*REAL(dp) :: thrfall*	Throughfall [m s-1]
in,out	*REAL(dp) :: snow_pack*	Snow pack [m]
out	*REAL(dp) :: deg_day*	Degree-day factor [m s-1 degreeC-1]
out	*REAL(dp) :: melt*	Melting snow depth [m s-1]
out	*REAL(dp) :: prec_effect*	Effective precipitation depth (snow melt + rain) [m]
out	*REAL(dp) :: rain*	Rain precipitation depth [m]
out	*REAL(dp) :: snow*	Snow precipitation depth [m]

Authors

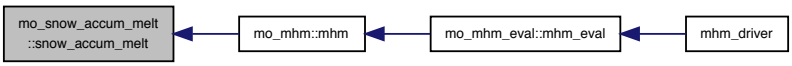
Vladyslav Prykhodko

Date

Dec 2012

Referenced by mo_mhm::mhm().

Here is the caller graph for this function:



16.89 mo_soil_database Module Reference

Generating soil database from input file.

Functions/Subroutines

- subroutine, public read_soil_lut (filename)

  Reads the soil LUT file.
- subroutine, public generate_soil_database

  Generates soil database.

16.89.1 Detailed Description

Generating soil database from input file.

This module provides the routines for generating the soil database for mHM from an ASCII input file. One routine *read_soil_LUT* reads a soil LookUpTable, performs some consistency checks and returns an initial soil database. The second routine *generate_soil_database* calculates based on the initial one the proper soil database.

Authors

Juliane Mai

Date

Dec 2012

16.89.2 Function/Subroutine Documentation

16.89.2.1 generate_soil_database()

```
subroutine, public mo_soil_database::generate_soil_database ( )
```

Generates soil database.

Calculates the proper soil database using the initialized soil database from read_soil_LUT.

Authors

Juliane Mai

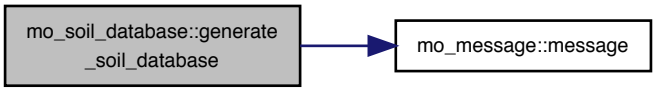
Date

Dec 2012

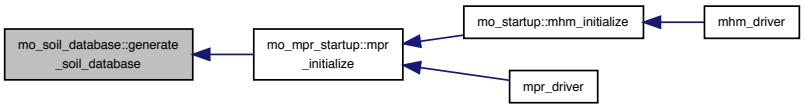
References mo_mpr_global_variables::horizondepth_mhm, mo_mpr_global_variables::iflag_soildb, mo_↩ message::message(), mo_common_constants::nodata_dp, mo_common_constants::nodata_i4, mo_mpr_global↩ _variables::nsoilhorizons_mhm, mo_mpr_global_variables::nsoiltypes, and mo_mpr_global_variables::soildb.

Referenced by mo_mpr_startup::mpr_initialize().

Here is the call graph for this function:



Here is the caller graph for this function:



16.89.2.2 read_soil_lut()

```
subroutine, public mo_soil_database::read_soil_lut (
             character(len = *), intent(in) filename )
```

Reads the soil LUT file.

Reads the soil LookUpTable file and checks for consistency.

Parameters

in	*character(len = *) :: filename*	filename of the soil LUT

Authors

Juliane Mai

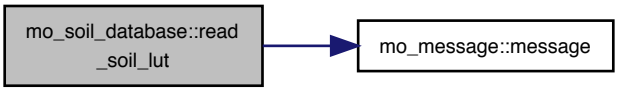
Date

Dec 2012

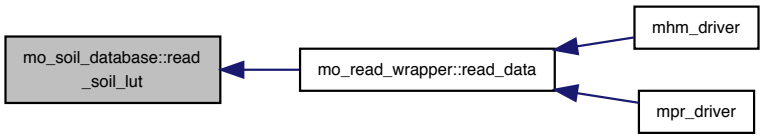
References mo_kind::dp, mo_common_constants::eps_dp, mo_mpr_global_variables::horizondepth_mhm, mo↩_mpr_global_variables::iflag_soildb, mo_message::message(), mo_mpr_constants::nlcover_class, mo_common↩_constants::nodata_dp, mo_common_constants::nodata_i4, mo_mpr_global_variables::nsoilhorizons_mhm, mo↩_mpr_global_variables::nsoiltypes, mo_mpr_global_variables::soildb, mo_mpr_global_variables::tillagedepth, and mo_mpr_file::usoil_database.

Referenced by mo_read_wrapper::read_data().

Here is the call graph for this function:



Here is the caller graph for this function:



16.90 mo_soil_moisture Module Reference

Soil moisture of the different layers.

Functions/Subroutines

- subroutine, public soil_moisture (processCase, frac_sealed, water_thresh_sealed, pet, evap_coeff, soil↩ _moist_sat, frac_roots, soil_moist_FC, wilting_point, soil_moist_exponen, jarvis_thresh_c1, aet_canopy, prec_effec, runoff_sealed, storage_sealed, infiltration, soil_moist, aet, aet_sealed)

  Soil moisture in different soil horizons.
- elemental pure real(dp) function, private feddes_et_reduction (soil_moist, soil_moist_FC, wilting_point, frac↩ _roots)

  stress factor for reducing evapotranspiration based on actual soil moisture
- elemental pure real(dp) function, private jarvis_et_reduction (soil_moist, soil_moist_sat, wilting_point, frac↩ _roots, jarvis_thresh_c1)

  stress factor for reducing evapotranspiration based on actual soil moisture

16.90.1 Detailed Description

Soil moisture of the different layers.

Soil moisture in the different layers is calculated with infiltration as $(\theta/\theta_{sat})^{\beta}$ Then evapotranspiration is calculated from PET with a soil water stress factor f_{SM} either using the Feddes equation - precessCase(1): $f_{SM} = \frac{\theta-\theta_{pwp}}{\theta_{fc}-\theta_{pwp}}$ or using the Jarvis equation - precessCase(1): $f_{SM} = \frac{1}{\theta_{stress-index-C1}} \frac{\theta-\theta_{pwp}}{\theta_{sat}-\theta_{pwp}}$.

Authors

Matthias Cuntz, Luis Samaniego

Date

Dec 2012

16.90.2 Function/Subroutine Documentation

16.90.2.1 feddes_et_reduction()

```
elemental pure real(dp) function, private mo_soil_moisture::feddes_et_reduction (
            real(dp), intent(in) soil_moist,
            real(dp), intent(in) soil_moist_FC,
            real(dp), intent(in) wilting_point,
            real(dp), intent(in) frac_roots )  [private]
```

stress factor for reducing evapotranspiration based on actual soil moisture

Potential evapotranspiration is reduced to 0 if SM is lower PWP. PET is equal fraction of roots if soil moisture is exceeding field capacity. If soil moisture is in between PWP and FC PET is reduced by fraction of roots times a stress factor. The ET reduction factor f is estimated as

$$f = \begin{cases} f_{roots} & if \theta \geq \theta_{fc} \\ f_{roots} \cdot \frac{\theta - \theta_{pwp}}{\theta_{fc} - \theta_{pwp}} & if \theta < \theta_{fc} \\ 0 & if \theta < \theta_{pwp} \end{cases}$$

Parameters

in	*real(dp) :: soil_moist*	Soil moisture of each horizon [mm]
in	*real(dp) :: soil_moist_FC*	Soil moisture below which actual ET is reduced [mm]
in	*real(dp) :: wilting_point*	Permanent wilting point
in	*real(dp) :: frac_roots*	Fraction of Roots in soil horizon is reduced [mm]

Returns

real(dp) :: feddes_et_reduction; et reduction factor

Authors

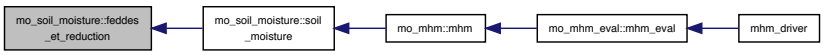
Matthias Cuntz, Cueneyd Demirel, Matthias Zink

Date

March 2017

Referenced by soil_moisture().

Here is the caller graph for this function:



16.90.2.2 jarvis_et_reduction()

```
elemental pure real(dp) function, private mo_soil_moisture::jarvis_et_reduction (
            real(dp), intent(in) soil_moist,
            real(dp), intent(in) soil_moist_sat,
```

```
            real(dp), intent(in) wilting_point,
            real(dp), intent(in) frac_roots,
            real(dp), intent(in) jarvis_thresh_c1 )  [private]
```

stress factor for reducing evapotranspiration based on actual soil moisture

The soil moisture stress factor is estimated based on the normalized soil water content. The normalized soil water content θ_{norm} is estimated as:

$$\theta_{norm} = \frac{\theta - \theta_{pwp}}{\theta_{sat} - \theta_{pwp}}$$

The ET reduction factor f is estimated as

$$f = \begin{cases} f_{roots} & if\, \theta_{norm} \geq jarvis\_sm\_threshold\_c1 \\ f_{roots} \frac{\theta_{norm}}{jarvis\_sm\_threshold\_c1} & if\, \theta_{norm} < jarvis\_sm\_threshold\_c1 \end{cases}$$

Parameters

in	*real(dp) :: soil_moist*	Soil moisture of each horizon [mm]
in	*real(dp) :: soil_moist_sat*	saturated Soil moisture content [mm]
in	*real(dp) :: wilting_point*	Permanent wilting point
in	*real(dp) :: frac_roots*	Fraction of Roots in soil horizon is reduced [mm]
in	*real(dp) :: jarvis_thresh_c1*	parameter C1 from Jarvis formulation

Returns

real(dp) :: jarvis_et_reduction; et reduction factor

Authors

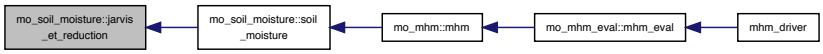
Cueneyd Demirel, Matthias Zink

Date

March 2017

Referenced by soil_moisture().

Here is the caller graph for this function:



16.90.2.3 soil_moisture()

```
subroutine, public mo_soil_moisture::soil_moisture (
            integer(i4), intent(in) processCase,
            real(dp), intent(in) frac_sealed,
            real(dp), intent(in) water_thresh_sealed,
            real(dp), intent(in) pet,
```

```
real(dp), intent(in) evap_coeff,
real(dp), dimension(:), intent(in) soil_moist_sat,
real(dp), dimension(:), intent(in) frac_roots,
real(dp), dimension(:), intent(in) soil_moist_FC,
real(dp), dimension(:), intent(in) wilting_point,
real(dp), dimension(:), intent(in) soil_moist_exponen,
real(dp), intent(in) jarvis_thresh_c1,
real(dp), intent(in) aet_canopy,
real(dp), intent(inout) prec_effec,
real(dp), intent(inout) runoff_sealed,
real(dp), intent(inout) storage_sealed,
real(dp), dimension(size(soil_moist_sat, 1)), intent(inout) infiltration,
real(dp), dimension(size(soil_moist_sat, 1)), intent(inout) soil_moist,
real(dp), dimension(size(soil_moist_sat, 1)), intent(out) aet,
real(dp), intent(out) aet_sealed )
```

Soil moisture in different soil horizons.

Infiltration I from one layer $k-1$ to the next k on pervious areas is calculated as (omit t)

$$I[k] = I[k-1](\theta[k]/\theta_{sat}[k])^{\beta[k]}$$

Then soil moisture can be calculated as (omit k)

$$\theta[t] = \theta[t-1] + I[t] - ET[t]$$

with ET (omit $[k,t]$) being

$$ET = f_{\text{roots}} \cdot f_{SM} \cdot PET$$

.

Parameters

in	*integer(i4) :: processCase*	1 - Feddes equation for PET reduction2 - Jarvis equation for PET reduction3 - Jarvis equation for PET reduction and FC dependency on root fraction coefficient
in	*real(dp) :: frac_sealed*	Fraction of sealed area
in	*real(dp) :: water_thresh_sealed*	Threshhold water depth in impervious areas [mm/s]
in	*real(dp) :: pet*	Reference evapotranspiration [mm/s]
in	*real(dp) :: evap_coeff*	Evaporation coefficent for free-water surface of that current month
in	*real(dp), dimension(:) :: soil_moist_sat*	Saturation soil moisture for each horizon [mm]
in	*real(dp), dimension(:) :: frac_roots*	Fraction of Roots in soil horizon
in	*real(dp), dimension(:) :: soil_moist_FC*	Soil moisture below which actual ET is reduced [mm]
in	*real(dp), dimension(:) :: wilting_point*	Permanent wilting point for each horizon [mm]
in	*real(dp), dimension(:) :: soil_moist_exponen*	Exponential parameter to how non-linear is the soil water retention
in	*real(dp) :: jarvis_thresh_c1*	Jarvis critical value for normalized soil water content
in	*real(dp) :: aet_canopy*	Actual ET from canopy [mm/s]
in,out	*real(dp) :: prec_effec*	Effective precipitation (rain + snow melt) [mm]
in,out	*real(dp) :: runoff_sealed*	Direct runoff from impervious areas
in,out	*real(dp) :: storage_sealed*	Retention storage of impervious areas
in,out	*real(dp), dimension(size(soil_moist_sat, 1)) :: infiltration*	Recharge, infiltration intensity oreffective precipitation of each horizon [mm/s]

Parameters

in,out	*real(dp), dimension(size(soil_moist_sat, 1)) :: soil_moist*	Soil moisture of each horizon [mm]
out	*real(dp), dimension(size(soil_moist_sat, 1)) :: aet*	actual ET [mm/s]
out	*real(dp) :: aet_sealed*	actual ET from free-water surfaces,i.e impervious cover [mm/s]

Authors

Matthias Cuntz

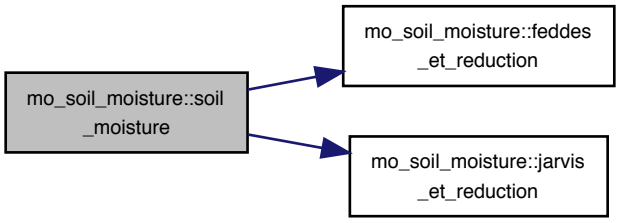
Date

Dec 2012

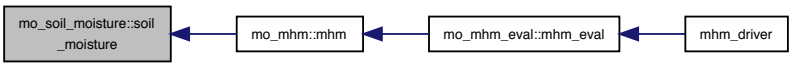
References mo_common_constants::eps_dp, feddes_et_reduction(), and jarvis_et_reduction().

Referenced by mo_mhm::mhm().

Here is the call graph for this function:



Here is the caller graph for this function:



16.91 mo_spatial_agg_disagg_forcing Module Reference

Spatial aggegation or disaggregation of meteorological input data.

Data Types

- interface spatial_aggregation

Spatial aggregation of meterological variables.

- interface spatial_disaggregation

  Spatial disaggregation of meterological variables.

Functions/Subroutines

- subroutine spatial_aggregation_3d (data2, cellsize2, cellsize1, mask1, mask2, data1)
- subroutine spatial_aggregation_4d (data2, cellsize2, cellsize1, mask1, mask2, data1)
- subroutine spatial_disaggregation_3d (data2, cellsize2, cellsize1, mask1, mask2, data1)
- subroutine spatial_disaggregation_4d (data2, cellsize2, cellsize1, mask1, mask2, data1)

16.91.1 Detailed Description

Spatial aggegation or disaggregation of meteorological input data.

This module contains two subroutines to upscale and downscale, respectively, the level-2 meterological inputs to a required Level-1 hydrological spatial resolution.

Authors

Rohini Kumar

Date

Jan 2013

16.91.2 Function/Subroutine Documentation

16.91.2.1 spatial_aggregation_3d()

```
subroutine mo_spatial_agg_disagg_forcing::spatial_aggregation_3d (
             real(dp), dimension(:, :, :), intent(in) data2,
             real(dp), intent(in) cellsize2,
             real(dp), intent(in) cellsize1,
             logical, dimension(:, :), intent(in) mask1,
             logical, dimension(:, :), intent(in) mask2,
             real(dp), dimension(:, :, :), intent(out), allocatable data1 )  [private]
```

References mo_common_constants::nodata_dp.

16.91.2.2 spatial_aggregation_4d()

```
subroutine mo_spatial_agg_disagg_forcing::spatial_aggregation_4d (
             real(dp), dimension(:, :, :, :), intent(in) data2,
             real(dp), intent(in) cellsize2,
             real(dp), intent(in) cellsize1,
             logical, dimension(:, :), intent(in) mask1,
             logical, dimension(:, :), intent(in) mask2,
             real(dp), dimension(:, :, :, :), intent(out), allocatable data1 )
```

References mo_common_constants::nodata_dp.

16.91.2.3 spatial_disaggregation_3d()

```
subroutine mo_spatial_agg_disagg_forcing::spatial_disaggregation_3d (
             real(dp), dimension(:, :, :), intent(in) data2,
             real(dp), intent(in) cellsize2,
             real(dp), intent(in) cellsize1,
             logical, dimension(:, :), intent(in) mask1,
             logical, dimension(:, :), intent(in) mask2,
             real(dp), dimension(:, :, :), intent(out), allocatable data1 )
```

References mo_common_constants::nodata_dp.

16.91.2.4 spatial_disaggregation_4d()

```
subroutine mo_spatial_agg_disagg_forcing::spatial_disaggregation_4d (
             real(dp), dimension(:, :, :, :), intent(in) data2,
             real(dp), intent(in) cellsize2,
             real(dp), intent(in) cellsize1,
             logical, dimension(:, :), intent(in) mask1,
             logical, dimension(:, :), intent(in) mask2,
             real(dp), dimension(:, :, :, :), intent(out), allocatable data1 )
```

References mo_common_constants::nodata_dp.

16.92 mo_spatialsimilarity Module Reference

Routines for bias insensitive comparison of spatial patterns.

Data Types

- interface nndv

  Calculates the number of neighboring dominating values, a measure for spatial dissimilarity.
- interface pd

  Calculates pattern dissimilarity (PD) measure.

Functions/Subroutines

- real(sp) function nndv_sp (mat1, mat2, mask, valid)
- real(dp) function nndv_dp (mat1, mat2, mask, valid)
- real(sp) function pd_sp (mat1, mat2, mask, valid)
- real(dp) function pd_dp (mat1, mat2, mask, valid)

16.92.1 Detailed Description

Routines for bias insensitive comparison of spatial patterns.

These routines are based on the idea that spatial similarity can be assessed by comparing the magnitude of neighboring pixels (e.g. is the neighboring pixel larger or smaller).

Author

Matthias Zink

Date

Mar 2013

16.92.2 Function/Subroutine Documentation

16.92.2.1 nndv_dp()

```
real(dp) function mo_spatialsimilarity::nndv_dp (
             real(dp), dimension(:, :), intent(in) mat1,
             real(dp), dimension(:, :), intent(in) mat2,
             logical, dimension(:, :), intent(in), optional mask,
             logical, intent(out), optional valid )
```

16.92.2.2 nndv_sp()

```
real(sp) function mo_spatialsimilarity::nndv_sp (
             real(sp), dimension(:, :), intent(in) mat1,
             real(sp), dimension(:, :), intent(in) mat2,
             logical, dimension(:, :), intent(in), optional mask,
             logical, intent(out), optional valid )
```

16.92.2.3 pd_dp()

```
real(dp) function mo_spatialsimilarity::pd_dp (
             real(dp), dimension(:, :), intent(in) mat1,
             real(dp), dimension(:, :), intent(in) mat2,
             logical, dimension(:, :), intent(in), optional mask,
             logical, intent(out), optional valid )
```

References mo_kind::dp.

16.92.2.4 pd_sp()

```
real(sp) function mo_spatialsimilarity::pd_sp (
             real(sp), dimension(:, :), intent(in) mat1,
             real(sp), dimension(:, :), intent(in) mat2,
             logical, dimension(:, :), intent(in), optional mask,
             logical, intent(out), optional valid )
```

References mo_kind::sp.

16.93 mo_standard_score Module Reference

Routines for calculating the normalization (anomaly)/standard score/z score and the deseasonalized (standard score on monthly basis) values of a time series.

Data Types

- interface classified_standard_score

  Calculates the classified standard score (e.g. classes are months).
- interface standard_score

  Calculates the standard score / normalization (anomaly) / z-score.

Functions/Subroutines

- real(sp) function, dimension(size(data, dim=1)) standard_score_sp (data, mask)
- real(dp) function, dimension(size(data, dim=1)) standard_score_dp (data, mask)
- real(sp) function, dimension(size(data, dim=1)) classified_standard_score_sp (data, classes, mask)
- real(dp) function, dimension(size(data, dim=1)) classified_standard_score_dp (data, classes, mask)

16.93.1 Detailed Description

Routines for calculating the normalization (anomaly)/standard score/z score and the deseasonalized (standard score on monthly basis) values of a time series.

In environmental research often the centralization and standardization are estimated for characterizing the dynamics of a signal.

Author

Matthias Zink

Date

May 2015

16.93.2 Function/Subroutine Documentation

16.93.2.1 classified_standard_score_dp()

```
real(dp) function, dimension(size(data, dim = 1)) mo_standard_score::classified_standard_↩
score_dp (
             real(dp), dimension(:), intent(in) data,
             integer, dimension(:), intent(in) classes,
             logical, dimension(:), intent(in), optional mask )
```

16.93.2.2 classified_standard_score_sp()

```
real(sp) function, dimension(size(data, dim = 1)) mo_standard_score::classified_standard_↩
score_sp (
             real(sp), dimension(:), intent(in) data,
             integer, dimension(:), intent(in) classes,
             logical, dimension(:), intent(in), optional mask )
```

16.93.2.3 standard_score_dp()

```
real(dp) function, dimension(size(data, dim = 1)) mo_standard_score::standard_score_dp (
             real(dp), dimension(:), intent(in) data,
             logical, dimension(:), intent(in), optional mask )
```

16.93.2.4 standard_score_sp()

```
real(sp) function, dimension(size(data, dim = 1)) mo_standard_score::standard_score_sp (
             real(sp), dimension(:), intent(in) data,
             logical, dimension(:), intent(in), optional mask )  [private]
```

16.94 mo_startup Module Reference

Startup procedures for mHM.

Functions/Subroutines

- subroutine, public mhm_initialize

  Initialize main mHM variables.
- subroutine constants_init

  Initialize mHM constants.
- subroutine l2_variable_init (iBasin, level0_iBasin, level2_iBasin)

  Initalize Level-2 meteorological forcings data.

16.94.1 Detailed Description

Startup procedures for mHM.

This module initializes all variables required to run mHM. This module needs to be run only one time at the beginning of a simulation if re-starting files do not exist.

Authors

Luis Samaniego, Rohini Kumar

Date

Dec 2012

16.94.2 Function/Subroutine Documentation

16.94.2.1 constants_init()

```
subroutine mo_startup::constants_init ( )
```

Initialize mHM constants.

transformation of time units & initialize constants

Authors

Luis Samaniego

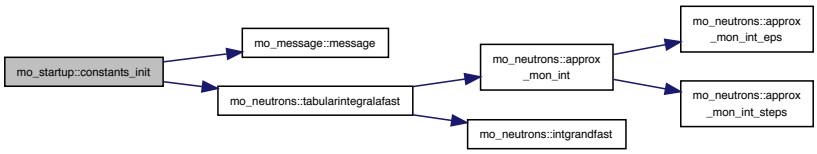
Date

Dec 2012

References mo_common_mhm_mrm_variables::c2tstu, mo_mpr_file::file_hydrogeoclass, mo_file::file_namelist↩
_mhm_param, mo_mpr_global_variables::geounitlist, mo_message::message(), mo_global_variables::neutron_↩
integral_afast, mo_common_variables::processmatrix, mo_neutrons::tabularintegralafast(), and mo_common_↩
mhm_mrm_variables::timestep.

Referenced by mhm_initialize().

Here is the call graph for this function:



Here is the caller graph for this function:

16.94.2.2 l2_variable_init()

```
subroutine mo_startup::l2_variable_init (
             integer(i4), intent(in) iBasin,
             type(grid), intent(in) level0_iBasin,
             type(grid), intent(inout) level2_iBasin )
```

Initalize Level-2 meteorological forcings data.

following tasks are performed 1) cell id & numbering 2) mask creation 3) append variable of intrest to global ones

Parameters

in	*integer(i4) :: iBasin*	Basin Id
in	*type(Grid) :: level0_iBasin*	
in,out	*type(Grid) :: level2_iBasin*	

Authors

Rohini Kumar

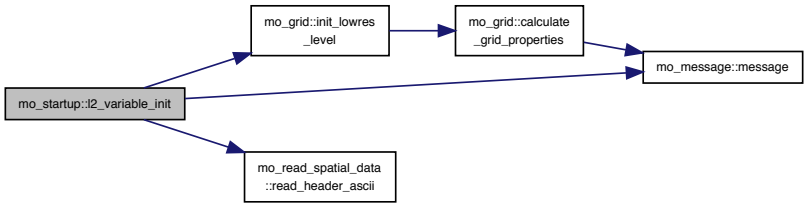
Date

Feb 2013

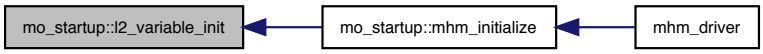
References mo_global_variables::dirprecipitation, mo_mpr_file::file_meteo_header, mo_grid::init_lowres_level(), mo_message::message(), mo_read_spatial_data::read_header_ascii(), and mo_mpr_file::umeteo_header.

Referenced by mhm_initialize().

Here is the call graph for this function:



Here is the caller graph for this function:



16.94.2.3 mhm_initialize()

```
subroutine, public mo_startup::mhm_initialize ( )
```

Initialize main mHM variables.

Initialize main mHM variables for a given basin. Calls the following procedures in this order:

- Constant initialization.
- Generate soil database.
- Checking inconsistencies input fields.
- Variable initialization at level-0.
- Variable initialization at level-1.
- Variable initialization at level-11.
- Space allocation of remaining variable/parameters. Global variables will be used at this stage.

Authors

Luis Samaniego, Rohini Kumar

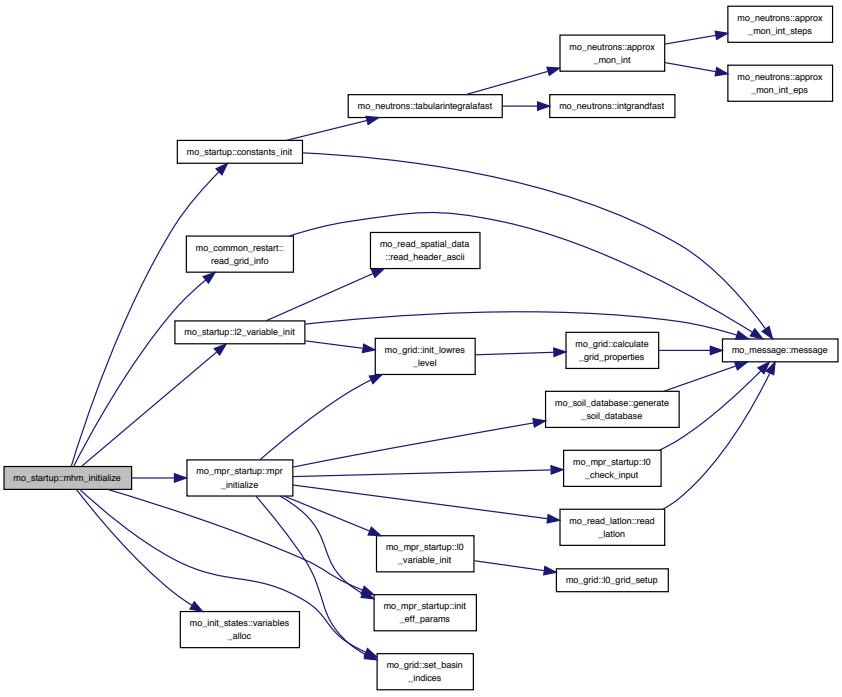
Date

Dec 2012

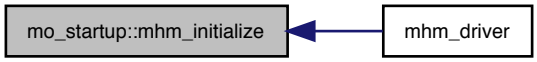
References constants_init(), mo_common_mhm_mrm_variables::dirrestartin, mo_kind::i4, mo_mpr_startup::init_↩ eff_params(), mo_common_variables::l0_basin, l2_variable_init(), mo_common_variables::level0, mo_common_↩ variables::level1, mo_global_variables::level2, mo_mpr_startup::mpr_initialize(), mo_common_variables::nbasins, mo_common_restart::read_grid_info(), mo_common_mhm_mrm_variables::read_restart, mo_grid::set_basin_↩ indices(), and mo_init_states::variables_alloc().

Referenced by mhm_driver().

Here is the call graph for this function:



Here is the caller graph for this function:



16.95 mo_string_utils Module Reference

String utilities.

Data Types

- interface num2str

  Convert to string.
- interface numarray2str

  Convert to string.

Functions/Subroutines

- character(len(whitespaces)) function, public compress (whiteSpaces, n)
- subroutine, public divide_string (string, delim, strArr)

  Divide string in substrings.
- logical function, public equalstrings (string1, string2)
- logical function, public nonull (str)

  Checks if string was already used.
- character(len=256) function, dimension(:), allocatable, public splitstring (string, delim)
- logical function, public startswith (string, start)
- character(len=len_trim(upper)) function, public tolower (upper)

  Convert to lower case.
- character(len=len_trim(lower)) function, public toupper (lower)
- pure character(len=10) function i42str (nn, form)
- pure character(len=20) function i82str (nn, form)
- pure character(len=32) function sp2str (rr, form)
- pure character(len=32) function dp2str (rr, form)
- pure character(len=10) function log2str (ll, form)
- character(len=size(arr)) function i4array2str (arr)
- integer(i4) function, dimension(:), allocatable, public str2num (string)

Variables

- character(len=∗), parameter, public separator = repeat('-', 70)

16.95.1 Detailed Description

String utilities.

This module provides string conversion and checking utilities.

Authors

Matthias Cuntz, Matthias Zink, Giovanni Dalmasso, David Schaefer

Date

Dec 2011

16.95.2 Function/Subroutine Documentation

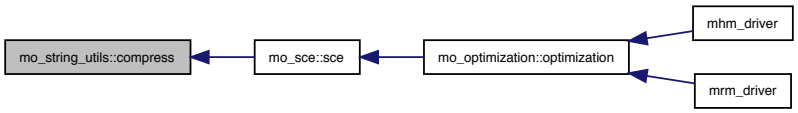
16.95.2.1 compress()

```
character(len(whitespaces)) function, public mo_string_utils::compress (
            character(len = *), intent(in) whiteSpaces,
            integer(i4), intent(out), optional n )
```

References mo_kind::i4.

Referenced by mo_sce::sce().

Here is the caller graph for this function:



16.95.2.2 divide_string()

```
subroutine, public mo_string_utils::divide_string (
            character(len = *), intent(in) string,
            character(len = *), intent(in) delim,
            character(len = *), dimension(:), intent(out), allocatable strArr )
```

Divide string in substrings.

Divides a string in several substrings (array of strings) with the help of a user specified delimiter.

Parameters

in	*CHARACTER(len=*), INTENT(IN) :: string*	- string to be divided
in	*CHARACTER(len=*), INTENT(IN) :: delim*	- delimiter specifying places for division
out	*CHARACTER(len=*), DIMENSION(:), ALLOCATABLE, INTENT(OUT) :: strArr*	Array of substrings, has to be allocateable and is handed to the routine unallocated

Author

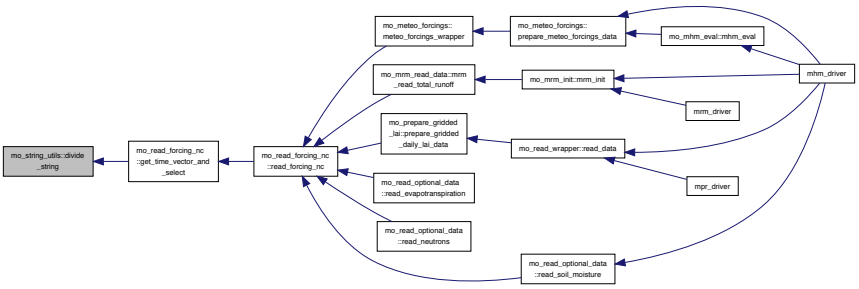
Matthias Zink

Date

Oct 2012

Referenced by mo_read_forcing_nc::get_time_vector_and_select().

Here is the caller graph for this function:



16.95.2.3 dp2str()

```
pure character(len = 32) function mo_string_utils::dp2str (
             real(dp), intent(in) rr,
             character(len = *), intent(in), optional form )  [private]
```

16.95.2.4 equalstrings()

```
logical function, public mo_string_utils::equalstrings (
             character(len = *), intent(in) string1,
             character(len = *), intent(in) string2 )
```

References str2num().

Here is the call graph for this function:

16.95.2.5 i42str()

```
pure character(len = 10) function mo_string_utils::i42str (
             integer(i4), intent(in) nn,
             character(len = *), intent(in), optional form )  [private]
```

16.95.2.6 i4array2str()

```
character(len = size(arr)) function mo_string_utils::i4array2str (
            integer(i4), dimension(:), intent(in) arr )  [private]
```

16.95.2.7 i82str()

```
pure character(len = 20) function mo_string_utils::i82str (
            integer(i8), intent(in) nn,
            character(len = *), intent(in), optional form )  [private]
```

16.95.2.8 log2str()

```
pure character(len = 10) function mo_string_utils::log2str (
            logical, intent(in) ll,
            character(len = *), intent(in), optional form )  [private]
```

16.95.2.9 nonull()

```
logical function, public mo_string_utils::nonull (
            character(len = *), intent(in) str )
```

Checks if string was already used.

Checks if string was already used, i.e. does not contain NULL character anymore.

Parameters

in	*character(len=*) :: str*	String

Returns

logical :: used — .true.: string was already set; .false.: string still in initialised state

Author

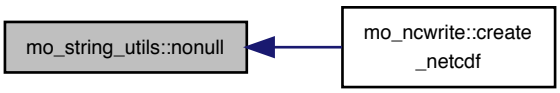
Matthias Cuntz

Date

Jan 2012

Referenced by mo_ncwrite::create_netcdf().

Here is the caller graph for this function:



16.95.2.10 sp2str()

```
pure character(len = 32) function mo_string_utils::sp2str (
             real(sp), intent(in) rr,
             character(len = *), intent(in), optional form )  [private]
```

16.95.2.11 splitstring()

```
character(len = 256) function, dimension(:), allocatable, public mo_string_utils::splitstring
(
             character(len = *), intent(in) string,
             character(len = *), intent(in) delim )
```

References str2num().

Here is the call graph for this function:

16.95.2.12 startswith()

```
logical function, public mo_string_utils::startswith (
             character(len = *), intent(in) string,
             character(len = *), intent(in) start )
```

References str2num().

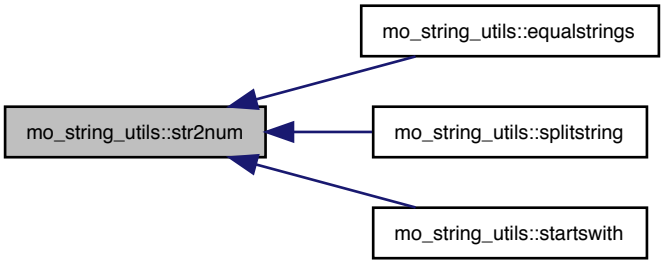
Here is the call graph for this function:

16.95.2.13 str2num()

```
integer(i4) function, dimension(:), allocatable, public mo_string_utils::str2num (
            character(len = *), intent(in) string )
```

Referenced by equalstrings(), splitstring(), and startswith().

Here is the caller graph for this function:



16.95.2.14 tolower()

```
character(len = len_trim(upper)) function, public mo_string_utils::tolower (
            character(len = *), intent(in) upper )
```

Convert to lower case.

Convert all upper case letters in string to lower case letters.

Parameters

in	*character(len=*) :: upper*	String

Returns

character(len=len_trim(upper)) :: low — String where all uppercase in input is converted to lowercase

Author

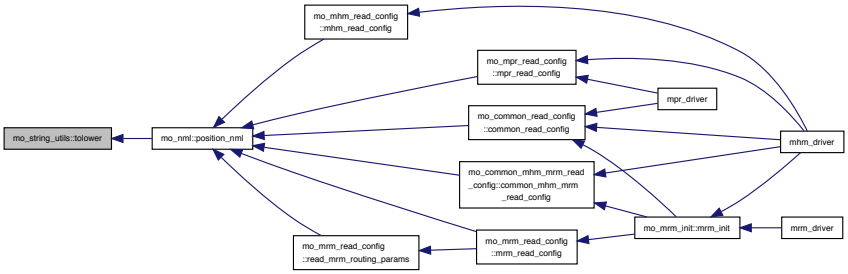
Matthias Cuntz - modified from Echam5, (C) MPI-MET, Hamburg, Germany

Date

Dec 2011

Referenced by mo_nml::position_nml().

Here is the caller graph for this function:

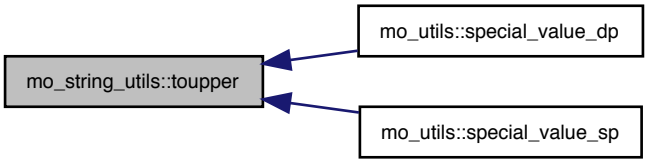


16.95.2.15 toupper()

```
character(len = len_trim(lower)) function, public mo_string_utils::toupper (
            character(len = *), intent(in) lower )
```

Referenced by mo_utils::special_value_dp(), and mo_utils::special_value_sp().

Here is the caller graph for this function:



16.95.3 Variable Documentation

16.95.3.1 separator

```
character(len = *), parameter, public mo_string_utils::separator = repeat('-', 70)
```

Referenced by mo_finish::finish(), mhm_driver(), and mo_mrm_init::print_startup_message().

16.96 mo_template Module Reference

Template for future module developments.

Data Types

- interface mean

  The average.

Functions/Subroutines

- elemental pure real(dp) function, public circum (radius)

  Circumference of a circle.
- real(dp) function mean_dp (dat, mask)
- real(sp) function mean_sp (dat, mask)

Variables

- real(dp), parameter, public pi_dp = 3.14159265358979323846264338327950288419 7_dp

  Constant Pi in double precision.
- real(sp), parameter, public pi_sp = 3.14159265358979323846264338327950288419 7_sp

  Constant Pi in single precision.
- integer(i4), parameter itest = 1

16.96.1 Detailed Description

Template for future module developments.

This module serves as a template for future model developments. It shows the module structure, the coding style, and documentation. Please read the Coding and documentation style guide.

Authors

Matthias Cuntz, Christoph Schneider

Date

Dec 2012

16.96.2 Function/Subroutine Documentation

16.96.2.1 circum()

```
elemental pure real(dp) function, public mo_template::circum (
            real(dp), intent(in) radius )
```

Circumference of a circle.

Calculates the circumference of a circle

$$c = 2\pi r$$

Parameters

in	*real(dp) :: radius*	Radius

Returns

real(dp) :: circum — circumference of circle.

Authors

Matthias Cuntz

Date

Dec 2012

References pi_dp.

16.96.2.2 mean_dp()

```
real(dp) function mo_template::mean_dp (
            real(dp), dimension(:), intent(in) dat,
            logical, dimension(:), intent(in), optional mask )  [private]
```

16.96.2.3 mean_sp()

```
real(sp) function mo_template::mean_sp (
            real(sp), dimension(:), intent(in) dat,
            logical, dimension(:), intent(in), optional mask )  [private]
```

16.96.3 Variable Documentation

16.96.3.1 itest

```
integer(i4), parameter mo_template::itest = 1  [private]
```

16.96.3.2 pi_dp

```
real(dp), parameter, public mo_template::pi_dp = 3.14159265358979323846264338327950288419 7_dp
```

Constant Pi in double precision.

Referenced by circum().

16.96.3.3 pi_sp

```
real(sp), parameter, public mo_template::pi_sp = 3.14159265358979323846264338327950288419 7_sp
```

Constant Pi in single precision.

16.97 mo_temporal_aggregation Module Reference

Temporal aggregation for time series (averaging)

Data Types

- interface day2mon_average

  Day-to-month average (day2mon_average)
- interface hour2day_average

  Hour-to-day average (hour2day_average)

Functions/Subroutines

- subroutine day2mon_average_dp (daily_data, yearS, monthS, dayS, mon_avg, misval, rm_misval)
- subroutine hour2day_average_dp (hourly_data, yearS, monthS, dayS, hourS, day_avg, misval, rm_misval)

16.97.1 Detailed Description

Temporal aggregation for time series (averaging)

This module does temporal aggregation (averaging) of time series

Authors

Oldrich Rakovec, Rohini Kumar

Date

October 2015

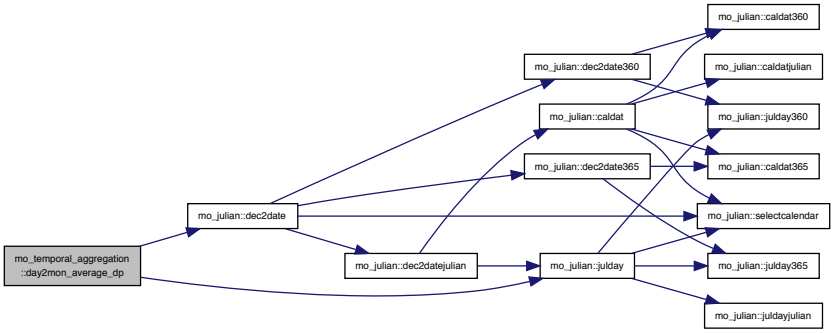
16.97.2 Function/Subroutine Documentation

16.97.2.1 day2mon_average_dp()

```
subroutine mo_temporal_aggregation::day2mon_average_dp (
            real(dp), dimension(:), intent(in) daily_data,
            integer(i4), intent(in) yearS,
            integer(i4), intent(in) monthS,
            integer(i4), intent(in) dayS,
            real(dp), dimension(:), intent(inout), allocatable mon_avg,
            real(dp), intent(in), optional misval,
            logical, intent(in), optional rm_misval )  [private]
```

References mo_julian::dec2date(), mo_common_constants::eps_dp, and mo_julian::julday().

Here is the call graph for this function:



16.97.2.2 hour2day_average_dp()

```
subroutine mo_temporal_aggregation::hour2day_average_dp (
            real(dp), dimension(:), intent(in) hourly_data,
            integer(i4), intent(in) yearS,
            integer(i4), intent(in) monthS,
            integer(i4), intent(in) dayS,
            integer(i4), intent(in) hourS,
            real(dp), dimension(:), intent(inout), allocatable day_avg,
            real(dp), intent(in), optional misval,
            logical, intent(in), optional rm_misval )  [private]
```

16.98 mo_temporal_disagg_forcing Module Reference

Temporal disaggregation of daily input values.

Functions/Subroutines

- elemental pure subroutine, public temporal_disagg_forcing (isday, ntimesteps_day, prec_day, pet_day, temp↩ _day, fday_prec, fday_pet, fday_temp, fnight_prec, fnight_pet, fnight_temp, temp_weights, pet_weights, pre↩ _weights, read_meteo_weights, prec, pet, temp)

  Temporally distribute daily mean forcings onto time step.

16.98.1 Detailed Description

Temporal disaggregation of daily input values.

Calculate actual values for precipitation, PET and temperature from daily mean inputs ote There is not PET correction for aspect in this routine. Use pet ∗ fasp before or after the routine.

Authors

Matthias Cuntz

Date

Dec 2012

16.98.2 Function/Subroutine Documentation

16.98.2.1 temporal_disagg_forcing()

```
elemental pure subroutine, public mo_temporal_disagg_forcing::temporal_disagg_forcing (
             logical, intent(in) isday,
             real(dp), intent(in) ntimesteps_day,
             real(dp), intent(in) prec_day,
             real(dp), intent(in) pet_day,
             real(dp), intent(in) temp_day,
             real(dp), intent(in) fday_prec,
             real(dp), intent(in) fday_pet,
             real(dp), intent(in) fday_temp,
             real(dp), intent(in) fnight_prec,
             real(dp), intent(in) fnight_pet,
             real(dp), intent(in) fnight_temp,
             real(dp), intent(in) temp_weights,
             real(dp), intent(in) pet_weights,
             real(dp), intent(in) pre_weights,
             logical, intent(in) read_meteo_weights,
             real(dp), intent(out) prec,
             real(dp), intent(out) pet,
             real(dp), intent(out) temp )
```

Temporally distribute daily mean forcings onto time step.

Calculates actual precipitation, PET and temperature from daily mean inputs. Precipitation and PET are distributed with predefined factors onto the day. Temperature gets a predefined amplitude added on day and substracted at night. Alternatively, weights for each hour and month can be given and disaggregation is using these as factors for PET and temperature. Precipitation is distributed uniformly.

Parameters

in	*logical :: isday*	is day or night
in	*real(dp) :: ntimesteps_day*	# of time steps per day
in	*real(dp) :: prec_day*	Daily mean precipitation [mm/s]
in	*real(dp) :: pet_day*	Daily mean ET [mm/s]
in	*real(dp) :: temp_day*	Daily mean air temperature [K]
in	*real(dp) :: fday_prec*	Daytime fraction of precipitation
in	*real(dp) :: fday_pet*	Daytime fraction of PET

Parameters

in	*real(dp) :: fday_temp*	Daytime air temparture increase
in	*real(dp) :: fnight_prec*	Daytime fraction of precipitation
in	*real(dp) :: fnight_pet*	Daytime fraction of PET
in	*real(dp) :: fnight_temp*	Daytime air temparture increase
in	*real(dp) :: temp_weights*	weights for average temperature
in	*real(dp) :: pet_weights*	weights for PET
in	*real(dp) :: pre_weights*	weights for precipitation
in	*logical :: read_meteo_weights*	flag indicating that weights should be used
out	*real(dp) :: prec*	Actual precipitation [mm/s]
out	*real(dp) :: pet*	Reference ET [mm/s]
out	*real(dp) :: temp*	Air temperature [K]

Authors

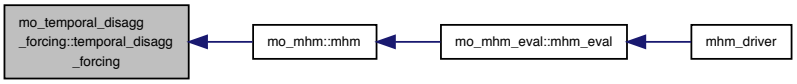
Matthias Cuntz

Date

Dec 2012

Referenced by mo_mhm::mhm().

Here is the caller graph for this function:



16.99 mo_timer Module Reference

Timing routines.

Functions/Subroutines

- subroutine, public timer_check (timer)

  Check a timer.
- subroutine, public timer_clear (timer)

  Reset a timer.
- real(sp) function, public timer_get (timer)

  Return a timer.
- subroutine, public timer_print (timer)

  Print a timer.
- subroutine, public timer_start (timer)

  Start a timer.
- subroutine, public timer_stop (timer)

Stop a timer.

- subroutine, public timers_init

  Initialise timer module.

Variables

- integer(i4), parameter, public max_timers = 99

  max number of timers allowed

- integer(i4), save, public cycles_max

  max value of clock allowed by system

- real(sp), save, public clock_rate

  clock_rate in seconds for each cycle

- integer(i4), dimension(max_timers), save, public cycles1

  cycle number at start for each timer

- integer(i4), dimension(max_timers), save, public cycles2

  cycle number at stop for each timer

- real(sp), dimension(max_timers), save, public cputime

  accumulated cpu time in each timer

- character(len=8), dimension(max_timers), save, public status

  timer status string

16.99.1 Detailed Description

Timing routines.

This module uses F90 cpu time routines to allowing setting of multiple CPU timers.

Authors

Matthias Cuntz - from timers.f (c) the Regents of the University of Californi

Date

Dec 2012

16.99.2 Function/Subroutine Documentation

16.99.2.1 timer_check()

```
subroutine, public mo_timer::timer_check (
             integer(i4), intent(in) timer )
```

Check a timer.

This routine checks a given timer. This is primarily used to periodically accumulate time in the timer to prevent timer cycles from wrapping around max_cycles.

Parameters

in	*integer(i4) :: timer*	timer number

Author

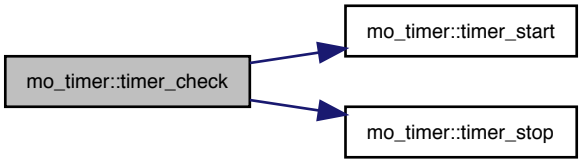
Matthias Cuntz

Date

Aug 2012

References status, timer_start(), and timer_stop().

Here is the call graph for this function:



16.99.2.2 timer_clear()

```
subroutine, public mo_timer::timer_clear (
             integer(i4), intent(in), optional timer )
```

Reset a timer.

This routine resets a given timer or all timers to 0.

Parameters

in	*integer(i4), optional :: timer*	timer number if given. If missing, all timers will be reset.

Author

Matthias Cuntz

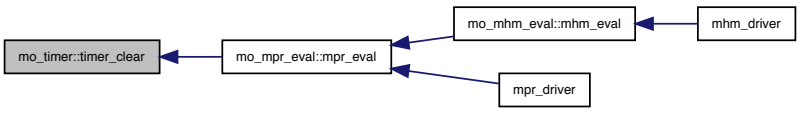
Date

Aug 2012

References cputime.

Referenced by mo_mpr_eval::mpr_eval().

Here is the caller graph for this function:



16.99.2.3 timer_get()

```
real(sp) function, public mo_timer::timer_get (
             integer(i4), intent(in) timer )
```

Return a timer.

This routine returns the result of a given timer. This can be called instead of timer_print so that the calling routine can print it in desired format.

Parameters

in	*integer(i4) :: timer*	timer number

Author

Matthias Cuntz

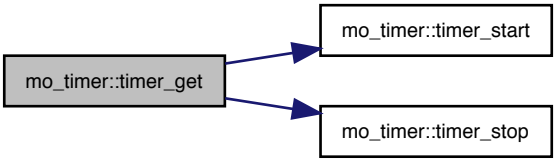
Date

Aug 2012

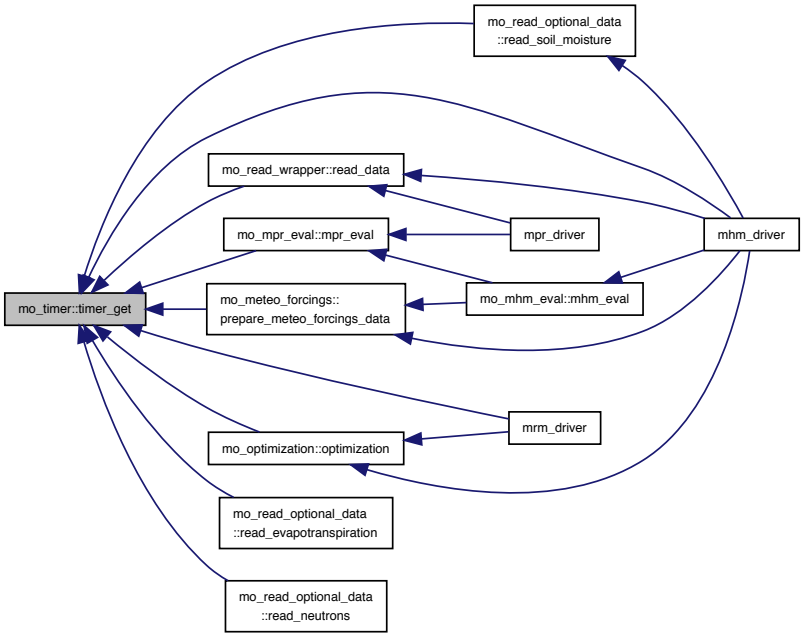
References cputime, status, timer_start(), and timer_stop().

Referenced by mhm_driver(), mo_mpr_eval::mpr_eval(), mrm_driver(), mo_optimization::optimization(), mo_meteo_forcings::prepare_meteo_forcings_data(), mo_read_wrapper::read_data(), mo_read_optional_data::read_evapotranspiration(), mo_read_optional_data::read_neutrons(), and mo_read_optional_data::read_soil_moisture().

Here is the call graph for this function:



Here is the caller graph for this function:



16.99.2.4 timer_print()

```
subroutine, public mo_timer::timer_print (
```

```
            integer(i4), intent(in) timer )
```

Print a timer.

This routine prints the accumulated cpu time in given timer.

Parameters

in	*integer(i4) :: timer*	timer number

Author

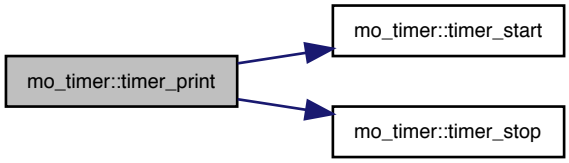
Matthias Cuntz

Date

Aug 2012

References cputime, status, timer_start(), and timer_stop().

Here is the call graph for this function:



16.99.2.5 timer_start()

```
subroutine, public mo_timer::timer_start (
            integer(i4), intent(in) timer )
```

Start a timer.

This routine starts a given timer.

Parameters

in	*integer(i4) :: timer*	timer number

Author

Matthias Cuntz

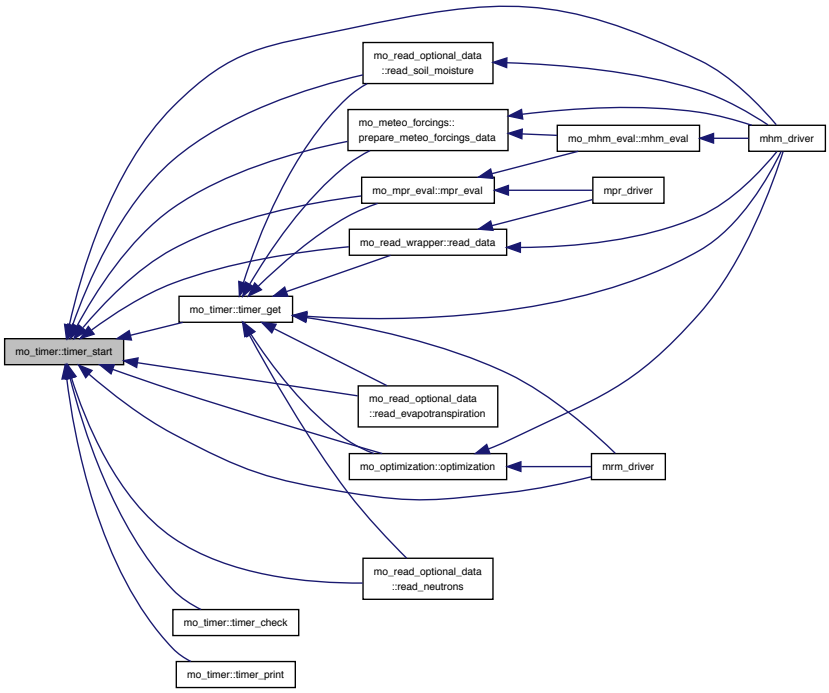
Date

Aug 2012

References cycles1, and status.

Referenced by mhm_driver(), mo_mpr_eval::mpr_eval(), mrm_driver(), mo_optimization::optimization(), mo_meteo_forcings::prepare_meteo_forcings_data(), mo_read_wrapper::read_data(), mo_read_optional_data::read_evapotranspiration(), mo_read_optional_data::read_neutrons(), mo_read_optional_data::read_soil_moisture(), timer_check(), timer_get(), and timer_print().

Here is the caller graph for this function:



16.99.2.6 timer_stop()

```
subroutine, public mo_timer::timer_stop (
             integer(i4), intent(in) timer )
```

Stop a timer.

This routine stops a given timer.

Parameters

in	*integer(i4) :: timer*	timer number

Author

Matthias Cuntz

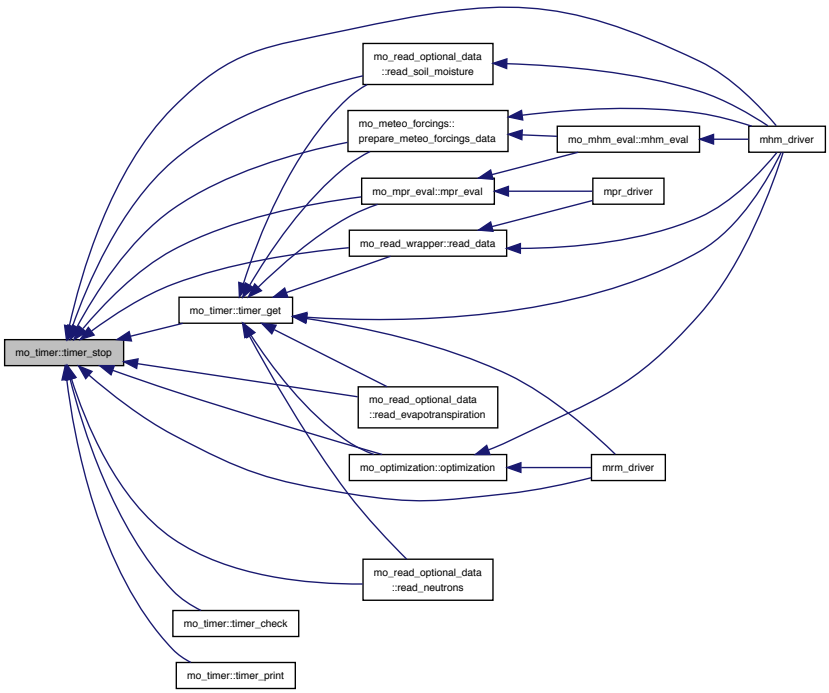
Date

Aug 2012

References clock_rate, cputime, cycles1, cycles2, cycles_max, and status.

Referenced by mhm_driver(), mo_mpr_eval::mpr_eval(), mrm_driver(), mo_optimization::optimization(), mo_meteo_forcings::prepare_meteo_forcings_data(), mo_read_wrapper::read_data(), mo_read_optional_data::read_evapotranspiration(), mo_read_optional_data::read_neutrons(), mo_read_optional_data::read_soil_moisture(), timer_check(), timer_get(), and timer_print().

Here is the caller graph for this function:



16.99.2.7 timers_init()

```
subroutine, public mo_timer::timers_init ( )
```

Initialise timer module.

This routine initializes some machine parameters necessary for computing cpu time from F90 intrinsics.

Author

Matthias Cuntz

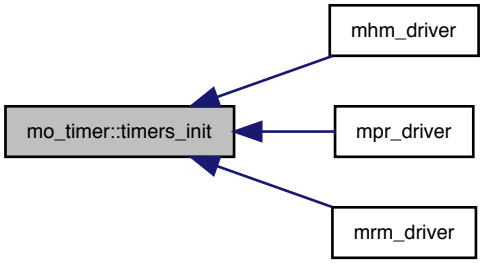
Date

Aug 2012

References clock_rate, cputime, cycles1, cycles2, cycles_max, and status.

Referenced by mhm_driver(), mpr_driver(), and mrm_driver().

Here is the caller graph for this function:



16.99.3 Variable Documentation

16.99.3.1 clock_rate

```
real(sp), save, public mo_timer::clock_rate
```

clock_rate in seconds for each cycle

Referenced by timer_stop(), and timers_init().

16.99.3.2 cputime

```
real(sp), dimension(max_timers), save, public mo_timer::cputime
```

accumulated cpu time in each timer

Referenced by timer_clear(), timer_get(), timer_print(), timer_stop(), and timers_init().

16.99.3.3 cycles1

```
integer(i4), dimension(max_timers), save, public mo_timer::cycles1
```

cycle number at start for each timer

Referenced by timer_start(), timer_stop(), and timers_init().

16.99.3.4 cycles2

```
integer(i4), dimension(max_timers), save, public mo_timer::cycles2
```

cycle number at stop for each timer

Referenced by timer_stop(), and timers_init().

16.99.3.5 cycles_max

```
integer(i4), save, public mo_timer::cycles_max
```

max value of clock allowed by system

Referenced by timer_stop(), and timers_init().

16.99.3.6 max_timers

```
integer(i4), parameter, public mo_timer::max_timers = 99
```

max number of timers allowed

16.99.3.7 status

```
character(len = 8), dimension(max_timers), save, public mo_timer::status
```

timer status string

Referenced by timer_check(), timer_get(), timer_print(), timer_start(), timer_stop(), and timers_init().

16.100 mo_upscaling_operators Module Reference

Module containing upscaling operators.

Functions/Subroutines

- integer(i4) function, dimension(size(l1_upper_rowid_cell, 1)), public majority_statistics (nClass, L1_upper_↩ rowId_cell, L1_lower_rowId_cell, L1_left_colonId_cell, L1_right_colonId_cell, L0_fineScale_2D_data)

  majority statistics
- real(dp) function, dimension(size(l0upbound_inlx, 1)), public l0_fractionalcover_in_lx (dataIn0, classId, mask0, L0upBound_inLx, L0downBound_inLx, L0leftBound_inLx, L0rightBound_inLx, nTCells0_inLx)

  fractional coverage of a given class of L0 fields in Lx field (Lx = L1 or L11)
- real(dp) function, dimension(size(nl0_cells_in_l1_cell, 1)), public upscale_arithmetic_mean (nL0_cells_in↩ _L1_cell, L1_upper_rowId_cell, L1_lower_rowId_cell, L1_left_colonId_cell, L1_right_colonId_cell, L0_cellId, mask0, nodata_value, L0_fineScale_data)

  aritmetic mean
- real(dp) function, dimension(size(nl0_cells_in_l1_cell, 1)), public upscale_harmonic_mean (nL0_cells_in↩ _L1_cell, L1_upper_rowId_cell, L1_lower_rowId_cell, L1_left_colonId_cell, L1_right_colonId_cell, L0_cellId, mask0, nodata_value, L0_fineScale_data)

  harmonic mean
- real(dp) function, dimension(size(l1_upper_rowid_cell, 1)), public upscale_geometric_mean (L1_upper_↩ rowId_cell, L1_lower_rowId_cell, L1_left_colonId_cell, L1_right_colonId_cell, mask0, nodata_value, L0_↩ fineScale_data)

  geometric mean
- real(dp) function, dimension(size(nl0_cells_in_l1_cell, 1)) upscale_p_norm (nL0_cells_in_L1_cell, L1↩ _upper_rowId_cell, L1_lower_rowId_cell, L1_left_colonId_cell, L1_right_colonId_cell, L0_cellId, mask0, nodata_value, p_norm, L0_fineScale_data)

  aritmetic mean

16.100.1 Detailed Description

Module containing upscaling operators.

This module provides the routines for upscaling_operators.

Authors

Giovanni Dalmasso, Rohini Kumar

Date

Dec 2012

16.100.2 Function/Subroutine Documentation

16.100.2.1 l0_fractionalcover_in_lx()

```
real(dp) function, dimension(size(l0upbound_inlx, 1)), public mo_upscaling_operators::l0_↩
fractionalcover_in_lx (
            integer(i4), dimension(:), intent(in) dataIn0,
            integer(i4), intent(in) classId,
            logical, dimension(:, :), intent(in) mask0,
            integer(i4), dimension(:), intent(in) L0upBound_inLx,
            integer(i4), dimension(:), intent(in) L0downBound_inLx,
            integer(i4), dimension(:), intent(in) L0leftBound_inLx,
            integer(i4), dimension(:), intent(in) L0rightBound_inLx,
            integer(i4), dimension(:), intent(in) nTCells0_inLx )
```

fractional coverage of a given class of L0 fields in Lx field (Lx = L1 or L11)

Fractional coverage of a given class of L0 fields in Lx field (Lx = L1 or L11). For example, this routine can be used for calculating the karstic fraction.

Parameters

in	*integer(i4), dimension(:) :: dataIn0*	input fields at finer scale
in	*integer(i4) :: classId*	class id for which fraction has to be estimated
in	*logical, dimension(:, :) :: mask0*	finer scale L0 mask
in	*integer(i4), dimension(:) :: L0upBound_inLx*	row start at finer L0 scale
in	*integer(i4), dimension(:) :: L0downBound_inLx*	row end at finer L0 scale
in	*integer(i4), dimension(:) :: L0leftBound_inLx*	col start at finer L0 scale
in	*integer(i4), dimension(:) :: L0rightBound_inLx*	col end at finer L0 scale
in	*integer(i4), dimension(:) :: nTCells0_inLx*	total number of valid L0 cells in a given Lx cell

Returns

real(dp) :: L0_fractionalCover_in_Lx(:) — packed 1D fraction coverage (Lx) of given class id

Authors

Rohini Kumar

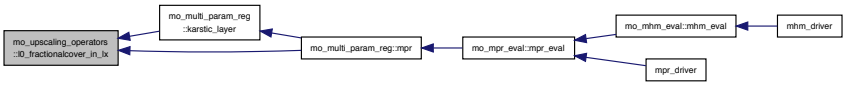
Date

Feb 2013

References mo_common_constants::nodata_i4.

Referenced by mo_multi_param_reg::karstic_layer(), and mo_multi_param_reg::mpr().

Here is the caller graph for this function:



16.100.2.2 majority_statistics()

```
integer(i4) function, dimension(size(l1_upper_rowid_cell, 1)), public mo_upscaling_operators↩
::majority_statistics (
            integer(i4), intent(in) nClass,
            integer(i4), dimension(:), intent(in) L1_upper_rowId_cell,
            integer(i4), dimension(:), intent(in) L1_lower_rowId_cell,
            integer(i4), dimension(:), intent(in) L1_left_colonId_cell,
            integer(i4), dimension(:), intent(in) L1_right_colonId_cell,
            integer(i4), dimension(:, :), intent(in) L0_fineScale_2D_data )
```

majority statistics

upscale grid L0_fineScale_2D_data based on a majority statistics

Parameters

in	*integer(i4) :: nClass*	number of classes
in	*integer(i4), dimension(:) :: L1_upper_rowId_cell*	upper row boundary (level-0) of a level-1 cell
in	*integer(i4), dimension(:) :: L1_lower_rowId_cell*	lower row boundary (level-0) of a level-1 cell
in	*integer(i4), dimension(:) :: L1_left_colonId_cell*	left colon boundary (level-0) of a level-1 cell
in	*integer(i4), dimension(:) :: L1_right_colonId_cell*	right colon boundary (level-0) of a level-1 cell
in	*integer(i4), dimension(:, :) :: L0_fineScale_2D_data*	high resolution data

Returns

integer(i4) :: majority_statistics(:) — Upscaled variable based on majority.

Authors

Giovanni Dalmasso, Rohini Kumar

Date

Dec 2012

16.100.2.3 upscale_arithmetic_mean()

```
real(dp) function, dimension(size(nl0_cells_in_l1_cell, 1)), public mo_upscaling_operators↩
::upscale_arithmetic_mean (
            integer(i4), dimension(:), intent(in) nL0_cells_in_L1_cell,
            integer(i4), dimension(:), intent(in) L1_upper_rowId_cell,
            integer(i4), dimension(:), intent(in) L1_lower_rowId_cell,
            integer(i4), dimension(:), intent(in) L1_left_colonId_cell,
            integer(i4), dimension(:), intent(in) L1_right_colonId_cell,
            integer(i4), dimension(:), intent(in) L0_cellId,
            logical, dimension(:, :), intent(in) mask0,
            real(dp), intent(in) nodata_value,
            real(dp), dimension(:), intent(in) L0_fineScale_data )
```

aritmetic mean

upscaling of level-0 grid data to level-1 using aritmetic mean

Parameters

in	*integer(i4), dimension(:) :: nL0_cells_in_L1_cell*	number of level-0 cells within a level-1 cell
in	*integer(i4), dimension(:) :: L1_upper_rowId_cell*	upper row boundary (level-0) of a level-1 cell
in	*integer(i4), dimension(:) :: L1_lower_rowId_cell*	lower row boundary (level-0) of a level-1 cell
in	*integer(i4), dimension(:) :: L1_left_colonId_cell*	left colon boundary (level-0) of a level-1 cell
in	*integer(i4), dimension(:) :: L1_right_colonId_cell*	right colon boundary (level-0) of a level-1 cell
in	*integer(i4), dimension(:) :: L0_cellId*	cell ID at level-0
in	*logical, dimension(:, :) :: mask0*	mask at level 0
in	*real(dp) :: nodata_value*	no data value
in	*real(dp), dimension(:) :: L0_fineScale_data*	high resolution data

Returns

real(dp) :: upscale_arithmetic_mean(:) — Upscaled variable from L0 to L1 using arithmetic mean

Authors

Giovanni Dalmasso, Rohini Kumar

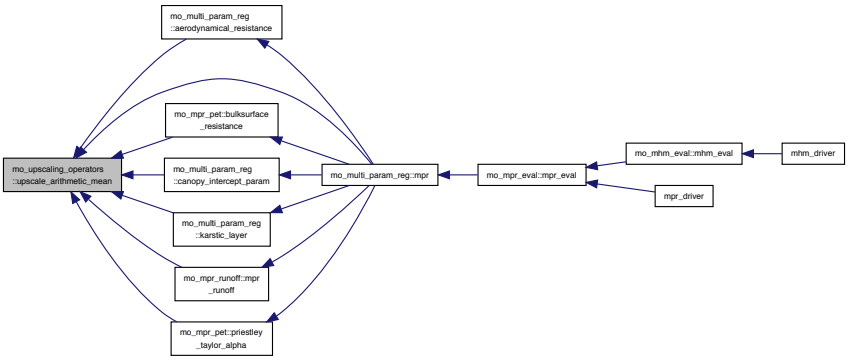
Date

Dec 2012

References mo_kind::i4.

Referenced by mo_multi_param_reg::aerodynamical_resistance(), mo_mpr_pet::bulksurface_resistance(), mo_multi_param_reg::canopy_intercept_param(), mo_multi_param_reg::karstic_layer(), mo_multi_param_reg::mpr(), mo_mpr_runoff::mpr_runoff(), and mo_mpr_pet::priestley_taylor_alpha().

Here is the caller graph for this function:



16.100.2.4 upscale_geometric_mean()

```
real(dp) function, dimension(size(l1_upper_rowid_cell, 1)), public mo_upscaling_operators↩
::upscale_geometric_mean (
            integer(i4), dimension(:), intent(in) L1_upper_rowId_cell,
            integer(i4), dimension(:), intent(in) L1_lower_rowId_cell,
            integer(i4), dimension(:), intent(in) L1_left_colonId_cell,
            integer(i4), dimension(:), intent(in) L1_right_colonId_cell,
            logical, dimension(:, :), intent(in) mask0,
            real(dp), intent(in) nodata_value,
            real(dp), dimension(:), intent(in) L0_fineScale_data )
```

geometric mean

upscaling of level-0 grid data to level-1 using geometric mean

Parameters

in	*integer(i4), dimension(:) :: L1_upper_rowId_cell*	upper row boundary (level-0) of a level-1 cell
in	*integer(i4), dimension(:) :: L1_lower_rowId_cell*	lower row boundary (level-0) of a level-1 cell
in	*integer(i4), dimension(:) :: L1_left_colonId_cell*	left colon boundary (level-0) of a level-1 cell
in	*integer(i4), dimension(:) :: L1_right_colonId_cell*	right colon boundary (level-0) of a level-1 cell
in	*logical, dimension(:, :) :: mask0*	mask at level 0
in	*real(dp) :: nodata_value*	no data value
in	*real(dp), dimension(:) :: L0_fineScale_data*	high resolution data

Returns

real(dp) :: upscale_geometric_mean(:) — Upscaled variable from L0 to L1 using geometric mean

Authors

Giovanni Dalmasso, Rohini Kumar

Date

Dec 2012

16.100.2.5 upscale_harmonic_mean()

```
real(dp) function, dimension(size(nl0_cells_in_l1_cell, 1)), public mo_upscaling_operators↩
::upscale_harmonic_mean (
             integer(i4), dimension(:), intent(in) nL0_cells_in_L1_cell,
             integer(i4), dimension(:), intent(in) L1_upper_rowId_cell,
             integer(i4), dimension(:), intent(in) L1_lower_rowId_cell,
             integer(i4), dimension(:), intent(in) L1_left_colonId_cell,
             integer(i4), dimension(:), intent(in) L1_right_colonId_cell,
             integer(i4), dimension(:), intent(in) L0_cellId,
             logical, dimension(:, :), intent(in) mask0,
             real(dp), intent(in) nodata_value,
             real(dp), dimension(:), intent(in) L0_fineScale_data )
```

harmonic mean

upscaling of level-0 grid data to level-1 using harmonic mean

Parameters

in	*integer(i4), dimension(:) :: nL0_cells_in_L1_cell*	number of level-0 cells within a level-1 cell
in	*integer(i4), dimension(:) :: L1_upper_rowId_cell*	upper row boundary (level-0) of a level-1 cell
in	*integer(i4), dimension(:) :: L1_lower_rowId_cell*	lower row boundary (level-0) of a level-1 cell
in	*integer(i4), dimension(:) :: L1_left_colonId_cell*	left colon boundary (level-0) of a level-1 cell
in	*integer(i4), dimension(:) :: L1_right_colonId_cell*	right colon boundary (level-0) of a level-1 cell
in	*integer(i4), dimension(:) :: L0_cellId*	cell ID at level-0
in	*logical, dimension(:, :) :: mask0*	mask at Level 0
in	*real(dp) :: nodata_value*	no data value
in	*real(dp), dimension(:) :: L0_fineScale_data*	high resolution data

Returns

real(dp) :: upscale_harmonic_mean(:) — Upscaled variable from L0 to L1 using harmonic mean

Authors

Giovanni Dalmasso, Rohini Kumar

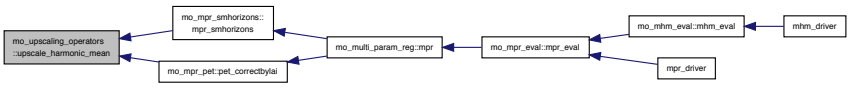
Date

Dec 2012

References mo_kind::i4.

Referenced by mo_mpr_smhorizons::mpr_smhorizons(), and mo_mpr_pet::pet_correctbylai().

Here is the caller graph for this function:



16.100.2.6 upscale_p_norm()

```
real(dp) function, dimension(size(nl0_cells_in_l1_cell, 1)) mo_upscaling_operators::upscale_p↩
_norm (
            integer(i4), dimension(:), intent(in) nL0_cells_in_L1_cell,
            integer(i4), dimension(:), intent(in) L1_upper_rowId_cell,
            integer(i4), dimension(:), intent(in) L1_lower_rowId_cell,
            integer(i4), dimension(:), intent(in) L1_left_colonId_cell,
            integer(i4), dimension(:), intent(in) L1_right_colonId_cell,
            integer(i4), dimension(:), intent(in) L0_cellId,
            logical, dimension(:, :), intent(in) mask0,
            real(dp), intent(in) nodata_value,
            real(dp), intent(in) p_norm,
            real(dp), dimension(:), intent(in) L0_fineScale_data )
```

aritmetic mean

upscaling of level-0 grid data to level-1 using aritmetic mean

Parameters

in	*integer(i4), dimension(:) :: nL0_cells_in_L1_cell*	number of level-0 cells within a level-1 cell
in	*integer(i4), dimension(:) :: L1_upper_rowId_cell*	upper row boundary (level-0) of a level-1 cell
in	*integer(i4), dimension(:) :: L1_lower_rowId_cell*	lower row boundary (level-0) of a level-1 cell
in	*integer(i4), dimension(:) :: L1_left_colonId_cell*	left colon boundary (level-0) of a level-1 cell
in	*integer(i4), dimension(:) :: L1_right_colonId_cell*	right colon boundary (level-0) of a level-1 cell
in	*integer(i4), dimension(:) :: L0_cellId*	cell ID at level-0
in	*logical, dimension(:, :) :: mask0*	mask at level 0
in	*real(dp) :: nodata_value*	no data value
in	*real(dp) :: p_norm*	p_norm value
in	*real(dp), dimension(:) :: L0_fineScale_data*	high resolution data

Returns

real(dp) :: upscale_arithmetic_mean(:) — Upscaled variable from L0 to L1 using arithmetic mean

Authors

Giovanni Dalmasso, Rohini Kumar

Date

Dec 2012

References mo_kind::i4.

16.101 mo_utils Module Reference

General utilities for the CHS library.

Data Types

- interface eq
- interface equal

  Comparison of real values.
- interface ge
- interface greaterequal
- interface is_finite

  .true. if not IEEE Inf, IEEE NaN, nor IEEE Inf nor IEEE NaN, respectively.
- interface is_nan
- interface is_normal
- interface le
- interface lesserequal
- interface locate

  Find closest values in a monotonic series, returns the indexes.
- interface ne
- interface notequal
- interface special_value

  Special IEEE values.
- interface swap

  Swap to values or two elements in array.

Functions/Subroutines

- elemental pure logical function equal_dp (a, b)
- elemental pure logical function equal_sp (a, b)
- elemental pure logical function greaterequal_dp (a, b)
- elemental pure logical function greaterequal_sp (a, b)
- elemental pure logical function lesserequal_dp (a, b)
- elemental pure logical function lesserequal_sp (a, b)
- elemental pure logical function notequal_dp (a, b)
- elemental pure logical function notequal_sp (a, b)
- elemental pure logical function is_finite_dp (a)
- elemental pure logical function is_finite_sp (a)
- elemental pure logical function is_nan_dp (a)
- elemental pure logical function is_nan_sp (a)
- elemental pure logical function is_normal_dp (a)
- elemental pure logical function is_normal_sp (a)

- integer(i4) function locate_0d_dp (x, y)
- integer(i4) function locate_0d_sp (x, y)
- integer(i4) function, dimension(:), allocatable locate_1d_dp (x, y)
- integer(i4) function, dimension(:), allocatable locate_1d_sp (x, y)
- elemental pure subroutine swap_xy_dp (x, y)
- elemental pure subroutine swap_xy_sp (x, y)
- elemental pure subroutine swap_xy_i4 (x, y)
- subroutine swap_vec_dp (x, i1, i2)
- subroutine swap_vec_sp (x, i1, i2)
- subroutine swap_vec_i4 (x, i1, i2)
- real(dp) function special_value_dp (x, ieee)
- real(sp) function special_value_sp (x, ieee)

16.101.1 Detailed Description

General utilities for the CHS library.

This module provides general utilities such as comparisons of two reals.

Authors

Matthias Cuntz, Juliane Mai

Date

Feb 2014

16.101.2 Function/Subroutine Documentation

16.101.2.1 equal_dp()

```
elemental pure logical function mo_utils::equal_dp (
             real(dp), intent(in) a,
             real(dp), intent(in) b )  [private]
```

16.101.2.2 equal_sp()

```
elemental pure logical function mo_utils::equal_sp (
             real(sp), intent(in) a,
             real(sp), intent(in) b )  [private]
```

16.101.2.3 greaterequal_dp()

```
elemental pure logical function mo_utils::greaterequal_dp (
             real(dp), intent(in) a,
             real(dp), intent(in) b )  [private]
```

16.101.2.4 greaterequal_sp()

```
elemental pure logical function mo_utils::greaterequal_sp (
            real(sp), intent(in) a,
            real(sp), intent(in) b )  [private]
```

16.101.2.5 is_finite_dp()

```
elemental pure logical function mo_utils::is_finite_dp (
            real(dp), intent(in) a )  [private]
```

16.101.2.6 is_finite_sp()

```
elemental pure logical function mo_utils::is_finite_sp (
            real(sp), intent(in) a )  [private]
```

16.101.2.7 is_nan_dp()

```
elemental pure logical function mo_utils::is_nan_dp (
            real(dp), intent(in) a )  [private]
```

16.101.2.8 is_nan_sp()

```
elemental pure logical function mo_utils::is_nan_sp (
            real(sp), intent(in) a )  [private]
```

16.101.2.9 is_normal_dp()

```
elemental pure logical function mo_utils::is_normal_dp (
            real(dp), intent(in) a )  [private]
```

16.101.2.10 is_normal_sp()

```
elemental pure logical function mo_utils::is_normal_sp (
            real(sp), intent(in) a )  [private]
```

16.101.2.11 lesserequal_dp()

```
elemental pure logical function mo_utils::lesserequal_dp (
            real(dp), intent(in) a,
            real(dp), intent(in) b )  [private]
```

16.101.2.12 lesserequal_sp()

```
elemental pure logical function mo_utils::lesserequal_sp (
             real(sp), intent(in) a,
             real(sp), intent(in) b )  [private]
```

16.101.2.13 locate_0d_dp()

```
integer(i4) function mo_utils::locate_0d_dp (
             real(dp), dimension(:), intent(in) x,
             real(dp), intent(in) y )  [private]
```

16.101.2.14 locate_0d_sp()

```
integer(i4) function mo_utils::locate_0d_sp (
             real(sp), dimension(:), intent(in) x,
             real(sp), intent(in) y )  [private]
```

16.101.2.15 locate_1d_dp()

```
integer(i4) function, dimension(:), allocatable mo_utils::locate_1d_dp (
             real(dp), dimension(:), intent(in) x,
             real(dp), dimension(:), intent(in) y )  [private]
```

16.101.2.16 locate_1d_sp()

```
integer(i4) function, dimension(:), allocatable mo_utils::locate_1d_sp (
             real(sp), dimension(:), intent(in) x,
             real(sp), dimension(:), intent(in) y )  [private]
```

16.101.2.17 notequal_dp()

```
elemental pure logical function mo_utils::notequal_dp (
             real(dp), intent(in) a,
             real(dp), intent(in) b )  [private]
```

16.101.2.18 notequal_sp()

```
elemental pure logical function mo_utils::notequal_sp (
             real(sp), intent(in) a,
             real(sp), intent(in) b )  [private]
```

16.101.2.19 special_value_dp()

```
real(dp) function mo_utils::special_value_dp (
            real(dp), intent(in) x,
            character(len = *), intent(in) ieee )  [private]
```

References mo_string_utils::toupper().

Here is the call graph for this function:

16.101.2.20 special_value_sp()

```
real(sp) function mo_utils::special_value_sp (
            real(sp), intent(in) x,
            character(len = *), intent(in) ieee )  [private]
```

References mo_string_utils::toupper().

Here is the call graph for this function:

16.101.2.21 swap_vec_dp()

```
subroutine mo_utils::swap_vec_dp (
            real(dp), dimension(:), intent(inout) x,
            integer(i4), intent(in) i1,
            integer(i4), intent(in) i2 )  [private]
```

16.101.2.22 swap_vec_i4()

```
subroutine mo_utils::swap_vec_i4 (
            integer(i4), dimension(:), intent(inout) x,
```

```
            integer(i4), intent(in) i1,
            integer(i4), intent(in) i2 )  [private]
```

16.101.2.23 swap_vec_sp()

```
subroutine mo_utils::swap_vec_sp (
            real(sp), dimension(:), intent(inout) x,
            integer(i4), intent(in) i1,
            integer(i4), intent(in) i2 )  [private]
```

16.101.2.24 swap_xy_dp()

```
elemental pure subroutine mo_utils::swap_xy_dp (
            real(dp), intent(inout) x,
            real(dp), intent(inout) y )  [private]
```

16.101.2.25 swap_xy_i4()

```
elemental pure subroutine mo_utils::swap_xy_i4 (
            integer(i4), intent(inout) x,
            integer(i4), intent(inout) y )  [private]
```

16.101.2.26 swap_xy_sp()

```
elemental pure subroutine mo_utils::swap_xy_sp (
            real(sp), intent(inout) x,
            real(sp), intent(inout) y )  [private]
```

16.102 mo_write_ascii Module Reference

Module to write ascii file output.

Functions/Subroutines

- subroutine, public write_configfile

  This modules writes the results of the configuration into an ASCII-file.
- subroutine, public write_optifile (best_OF, best_paramSet, param_names)

  Write briefly final optimization results.
- subroutine, public write_optinamelist (processMatrix, parameters, maskpara, parameters_name)

  Write final, optimized parameter set in a namelist format.

16.102.1 Detailed Description

Module to write ascii file output.

Module to write ascii file output. Writing model output to ASCII should be the exception. Therefore, output is written usually as NetCDF and only: (1) The configuration file of mHM, (2) the final parameter set after optimization, and (3) the simulated vs. observed daily discharge is written in ASCII file format to allow for a quick assurance of proper model runs.

Authors

Christoph Schneider, Juliane Mai, Luis Samaniego

Date

May 2013

16.102.2 Function/Subroutine Documentation

16.102.2.1 write_configfile()

```
subroutine, public mo_write_ascii::write_configfile ( )
```

This modules writes the results of the configuration into an ASCII-file.

TODO: add description

Authors

Christoph Schneider

Date

May 2013 TODO: add description

TODO: add description

Authors

Robert Schweppe

Date

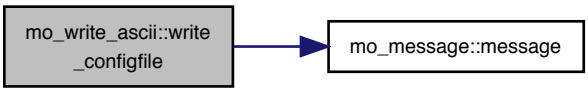
Jun 2018

References mo_common_variables::dirconfigout, mo_mrm_global_variables::dirgauges, mo_common_↩ variables::dirlcover, mo_common_variables::dirmorpho, mo_common_variables::dirout, mo_global_variables↩ ::dirprecipitation, mo_global_variables::dirreferenceet, mo_common_variables::dirrestartout, mo_global_↩ variables::dirtemperature, mo_common_mhm_mrm_variables::evalper, mo_common_file::file_config, mo_↩ mrm_global_variables::gauge, mo_common_variables::global_parameters, mo_common_variables::global↩ _parameters_name, mo_kind::i4, mo_common_variables::iflag_cordinate_sys, mo_mrm_global_variables↩ ::inflowgauge, mo_common_variables::l0_basin, mo_mrm_global_variables::l11_fromn, mo_mrm_global_↩ variables::l11_label, mo_mrm_global_variables::l11_length, mo_mrm_global_variables::l11_netperm, mo_mrm↩ _global_variables::l11_noutlets, mo_mrm_global_variables::l11_rorder, mo_mrm_global_variables::l11_slope, mo_mrm_global_variables::l11_ton, mo_mrm_global_variables::l1_l11_id, mo_common_variables::lc_year_end, mo_common_variables::lc_year_start, mo_common_variables::lcfilename, mo_common_mhm_mrm_variables↩ ::lcyearid, mo_common_variables::level0, mo_common_variables::level1, mo_mrm_global_variables::level11,

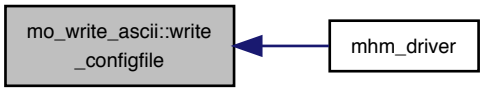
mo_message::message(), mo_common_variables::nbasins, mo_mrm_global_variables::ngaugestotal, mo_mrm_global_variables::ninflowgaugestotal, mo_common_variables::nlcoverscene, mo_common_constants::nodata_dp, mo_common_variables::processmatrix, mo_common_mhm_mrm_variables::read_restart, mo_common_variables::resolutionhydrology, mo_common_mhm_mrm_variables::resolutionrouting, mo_common_mhm_mrm_variables::simper, mo_common_mhm_mrm_variables::timestep, mo_common_file::uconfig, mo_file::version, mo_common_mhm_mrm_variables::warmper, and mo_common_variables::write_restart.

Referenced by mhm_driver().

Here is the call graph for this function:



Here is the caller graph for this function:



16.102.2.2 write_optifile()

```
subroutine, public mo_write_ascii::write_optifile (
             real(dp), intent(in) best_OF,
             real(dp), dimension(:), intent(in) best_paramSet,
             character(len = *), dimension(:), intent(in) param_names )
```

Write briefly final optimization results.

Write overall best objective function and the best optimized parameter set to a file_opti.

Parameters

in	*real(dp) :: best_OF*	best objective function value as returnedby the optimization routine
in	*real(dp), dimension(:) :: best_paramSet*	best associated global parameter setCalled only when optimize is .TRUE.
in	*character(len = *), dimension(:) :: param_names*	

Authors

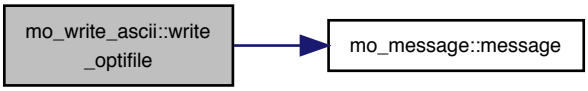
David Schaefer

Date

July 2013

References mo_common_variables::dirconfigout, mo_common_mhm_mrm_file::file_opti, mo_message::message(), and mo_common_mhm_mrm_file::uopti.

Here is the call graph for this function:



16.102.2.3 write_optinamelist()

```
subroutine, public mo_write_ascii::write_optinamelist (
             integer(i4), dimension(nprocesses, 3), intent(in) processMatrix,
             real(dp), dimension(:, :), intent(in) parameters,
             logical, dimension(size(parameters, 1)), intent(in) maskpara,
             character(len = *), dimension(size(parameters, 1)), intent(in) parameters_name )
```

Write final, optimized parameter set in a namelist format.

Write final, optimized parameter set in a namelist format. Only parameters of processes which were switched on are written to the namelist. All others are discarded.

Parameters

in	*integer(i4), dimension(nProcesses, 3) :: processMatrix*	information about which process case was used
in	*real(dp), dimension(:, :) :: parameters*	(min, max, opti)
in	*logical, dimension(size(parameters, 1)) :: maskpara*	.true. if parameter was calibrated
in	*character(len = *), dimension(size(parameters, 1)) :: parameters_name*	clear names of parameters

Authors

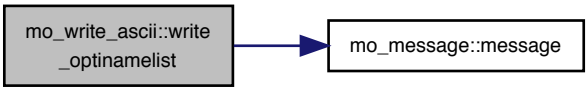
Juliane Mai

Date

Dec 2013

References mo_common_variables::dirconfigout, mo_common_mhm_mrm_file::file_opti_nml, mo_message::message(), mo_common_variables::nprocesses, and mo_common_mhm_mrm_file::uopti_nml.

Here is the call graph for this function:



16.103 mo_write_fluxes_states Module Reference

Creates NetCDF output for different fluxes and state variables of mHM.

Data Types

- interface outputdataset
- interface outputvariable

Functions/Subroutines

- type(outputvariable) function newoutputvariable (nc, name, dtype, dims, ncells, mask, avg)

  Initialize OutputVariable.
- subroutine updatevariable (self, data)

  Update OutputVariable.
- subroutine writevariabletimestep (self, timestep)

  Write timestep to file.
- type(outputdataset) function, public newoutputdataset (ibasin, mask1, nCells)

  Initialize OutputDataset.
- subroutine updatedataset (self, sidx, eidx, L1_fSealed, L1_fNotSealed, L1_inter, L1_snowPack, L1_soilMoist, L1_soilMoistSat, L1_sealSTW, L1_unsatSTW, L1_satSTW, L1_neutrons, L1_pet, L1_aETSoil, L1_aETCanopy, L1_aETSealed, L1_total_runoff, L1_runoffSeal, L1_fastRunoff, L1_slowRunoff, L1_baseflow, L1_percol, L1_infilSoil, L1_preEffect)

  Update all variables.
- subroutine writetimestep (self, timestep)

  Write all accumulated data.
- subroutine close (self)

  Close the file.
- type(ncdataset) function createoutputfile (ibasin)

  Create and initialize output file.
- subroutine writevariableattributes (var, long_name, unit)

  Write output variable attributes.
- character(16) function fluxesunit (ibasin)

  Generate a unit string.

16.103.1 Detailed Description

Creates NetCDF output for different fluxes and state variables of mHM.

NetCDF is first initialized and later on variables are put to the NetCDF.

Authors

Matthias Zink

Date

Apr 2013

16.103.2 Function/Subroutine Documentation

16.103.2.1 close()

```
subroutine mo_write_fluxes_states::close (
            class(outputdataset) self )  [private]
```

Close the file.

Close the file associated with variable of type(OutputDataset)

Authors

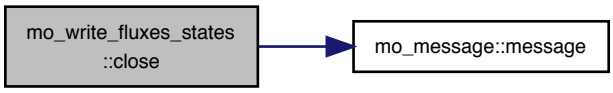
Rohini Kumar & Stephan Thober

Date

August 2013

References mo_common_variables::dirout, and mo_message::message().

Here is the call graph for this function:



16.103.2.2 createoutputfile()

```
type(ncdataset) function mo_write_fluxes_states::createoutputfile (
            integer(i4), intent(in) ibasin )
```

Create and initialize output file.

Create output file, write all non-dynamic variables and global attributes for the given basin.

Returns

type(NcDataset)

Parameters

in	*integer(i4) :: ibasin*	-> basin id

Authors

David Schaefer

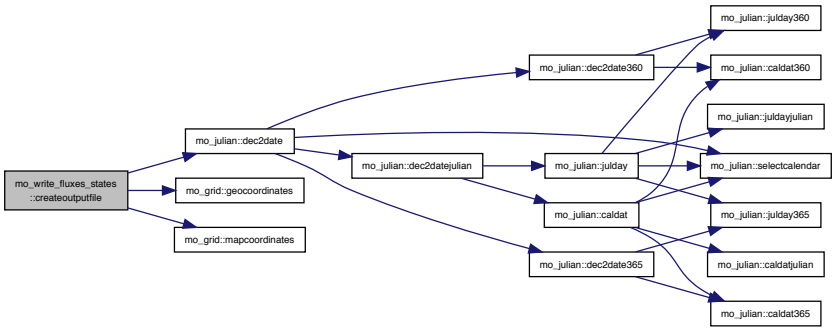
Date

June 2015

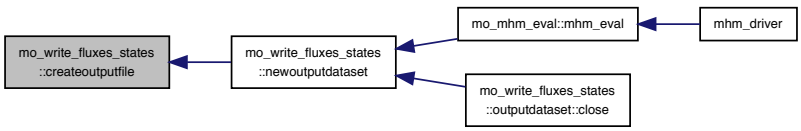
References mo_julian::dec2date(), mo_common_variables::dirout, mo_common_mhm_mrm_variables::evalper, mo_grid::geocoordinates(), mo_common_variables::level1, mo_grid::mapcoordinates(), mo_common_constants↩ ::nodata_dp, and mo_file::version.

Referenced by newoutputdataset().

Here is the call graph for this function:



Here is the caller graph for this function:



16.103.2.3 fluxesunit()

```
character(16) function mo_write_fluxes_states::fluxesunit (
            integer(i4), intent(in) ibasin )  [private]
```

Generate a unit string.

Generate the unit string for the output variable netcdf attribute based on modeling timestep

Returns

character(16)

Parameters

in	*integer(i4) :: ibasin*	

Authors

David Schaefer

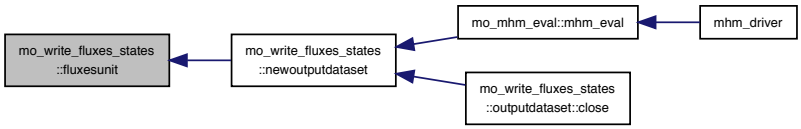
Date

June 2015

References mo_common_mhm_mrm_variables::ntstepday, mo_common_mhm_mrm_variables::simper, mo_common_mhm_mrm_variables::timestep, and mo_global_variables::timestep_model_outputs.

Referenced by newoutputdataset().

Here is the caller graph for this function:



16.103.2.4 newoutputdataset()

```
type(outputdataset) function, public mo_write_fluxes_states::newoutputdataset (
            integer(i4), intent(in) ibasin,
            logical, dimension(:, :), intent(in), pointer mask1,
            integer(i4), intent(in) nCells )
```

Initialize OutputDataset.

Create and initialize the output file. If new a new output variable needs to be written, this is the first of two procedures to change (second: updateDataset)

Returns

type(OutputDataset)

Parameters

in	*integer(i4) :: ibasin*	-> basin id
in	*logical, dimension(:, :) :: mask1*	-> L1 mask to reconstruct the data
in	*integer(i4) :: nCells*	

Authors

Matthias Zink

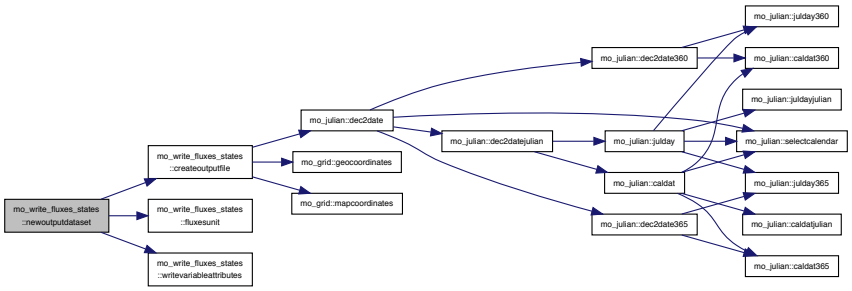
Date

Apr 2013

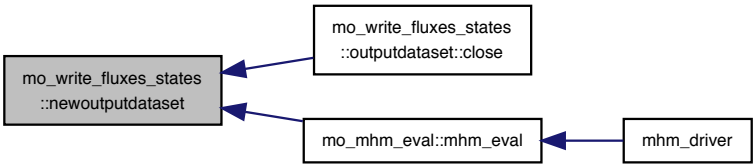
References createoutputfile(), fluxesunit(), mo_mpr_global_variables::nsoilhorizons_mhm, mo_global_variables↩ ::outputflxstate, and writevariableattributes().

Referenced by mo_write_fluxes_states::outputdataset::close(), and mo_mhm_eval::mhm_eval().

Here is the call graph for this function:



Here is the caller graph for this function:



16.103.2.5 newoutputvariable()

```
type(outputvariable) function mo_write_fluxes_states::newoutputvariable (
            type(ncdataset), intent(in) nc,
            character(*), intent(in) name,
            character(*), intent(in) dtype,
            character(16), dimension(3), intent(in) dims,
            integer(i4), intent(in) ncells,
            logical, dimension(:, :), intent(in), target mask,
            logical, intent(in), optional avg )  [private]
```

Initialize OutputVariable.

TODO: add description

Returns

type(OutputVariable)

Parameters

in	*type(NcDataset) :: nc*	-> NcDataset which contains the variable
in	*character(∗) :: name*	
in	*character(∗) :: dtype*	
in	*character(16), dimension(3) :: dims*	
in	*integer(i4) :: ncells*	-> number of cells in basin
in	*logical, dimension(:, :) :: mask*	
in	*logical, optional :: avg*	-> average the data before writing

Authors

David Schaefer

Date

June 2015

16.103.2.6 updatedataset()

```
subroutine mo_write_fluxes_states::updatedataset (
             class(outputdataset), intent(inout), target self,
             integer(i4), intent(in) sidx,
             integer(i4), intent(in) eidx,
             real(dp), dimension(:), intent(in) L1_fSealed,
             real(dp), dimension(:), intent(in) L1_fNotSealed,
             real(dp), dimension(:), intent(in) L1_inter,
             real(dp), dimension(:), intent(in) L1_snowPack,
             real(dp), dimension(:, :), intent(in) L1_soilMoist,
             real(dp), dimension(:, :), intent(in) L1_soilMoistSat,
             real(dp), dimension(:), intent(in) L1_sealSTW,
             real(dp), dimension(:), intent(in) L1_unsatSTW,
             real(dp), dimension(:), intent(in) L1_satSTW,
             real(dp), dimension(:), intent(in) L1_neutrons,
             real(dp), dimension(:), intent(in) L1_pet,
             real(dp), dimension(:, :), intent(in) L1_aETSoil,
             real(dp), dimension(:), intent(in) L1_aETCanopy,
             real(dp), dimension(:), intent(in) L1_aETSealed,
             real(dp), dimension(:), intent(in) L1_total_runoff,
             real(dp), dimension(:), intent(in) L1_runoffSeal,
             real(dp), dimension(:), intent(in) L1_fastRunoff,
             real(dp), dimension(:), intent(in) L1_slowRunoff,
             real(dp), dimension(:), intent(in) L1_baseflow,
             real(dp), dimension(:), intent(in) L1_percol,
             real(dp), dimension(:, :), intent(in) L1_infilSoil,
             real(dp), dimension(:), intent(in) L1_preEffect )
```

Update all variables.

Call the type bound procedure updateVariable for all output variables. If a new output variable needs to be written, this is the second of two procedures to change (first: newOutputDataset)

Parameters

in,out	*class(OutputDataset) :: self*	
in	*integer(i4) :: sidx, eidx*	
in	*integer(i4) :: sidx, eidx*	
in	*real(dp), dimension(:) :: L1_fSealed*	
in	*real(dp), dimension(:) :: L1_fNotSealed*	
in	*real(dp), dimension(:) :: L1_inter*	
in	*real(dp), dimension(:) :: L1_snowPack*	
in	*real(dp), dimension(:, :) :: L1_soilMoist*	
in	*real(dp), dimension(:, :) :: L1_soilMoistSat*	
in	*real(dp), dimension(:) :: L1_sealSTW*	
in	*real(dp), dimension(:) :: L1_unsatSTW*	
in	*real(dp), dimension(:) :: L1_satSTW*	
in	*real(dp), dimension(:) :: L1_neutrons*	
in	*real(dp), dimension(:) :: L1_pet*	
in	*real(dp), dimension(:, :) :: L1_aETSoil*	
in	*real(dp), dimension(:) :: L1_aETCanopy*	
in	*real(dp), dimension(:) :: L1_aETSealed*	
in	*real(dp), dimension(:) :: L1_total_runoff*	
in	*real(dp), dimension(:) :: L1_runoffSeal*	
in	*real(dp), dimension(:) :: L1_fastRunoff*	
in	*real(dp), dimension(:) :: L1_slowRunoff*	
in	*real(dp), dimension(:) :: L1_baseflow*	
in	*real(dp), dimension(:) :: L1_percol*	
in	*real(dp), dimension(:, :) :: L1_infilSoil*	
in	*real(dp), dimension(:) :: L1_preEffect*	

Authors

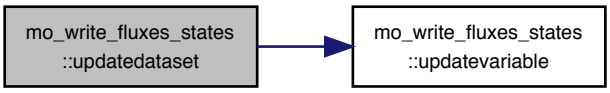
Matthias Zink

Date

Apr 2013

References mo_mpr_global_variables::nsoilhorizons_mhm, mo_global_variables::outputflxstate, and updatevariable().

Here is the call graph for this function:



16.103.2.7 updatevariable()

```
subroutine mo_write_fluxes_states::updatevariable (
            class(outputvariable), intent(inout) self,
            real(dp), dimension(:), intent(in) data )  [private]
```

Update OutputVariable.

Add the array given as actual argument to the derived type's component 'data'

Returns

type(OutputVariable)

Parameters

in,out	*class(OutputVariable) :: self*	
in	*real(dp), dimension(:) :: data*	

Authors

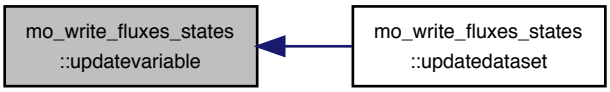
David Schaefer

Date

June 2015

Referenced by updatedataset().

Here is the caller graph for this function:



16.103.2.8 writetimestep()

```
subroutine mo_write_fluxes_states::writetimestep (
            class(outputdataset), intent(inout), target self,
            integer(i4), intent(in) timestep )
```

Write all accumulated data.

Write all accumulated and potentially averaged data to disk.

Returns

type(OutputVariable)

Parameters

in,out	*class(OutputDataset) :: self*	
in	*integer(i4) :: timestep*	The model timestep to write

Authors

David Schaefer

Date

June 2015

16.103.2.9 writevariableattributes()

```
subroutine mo_write_fluxes_states::writevariableattributes (
             type(outputvariable), intent(in) var,
             character(*), intent(in) long_name,
             character(*), intent(in) unit )
```

Write output variable attributes.

TODO: add description

Parameters

in	*type(OutputVariable) :: var*	
in	*character(*) :: long_name, unit*	-> variable name
in	*character(*) :: long_name, unit*	-> physical unit

Authors

David Schaefer

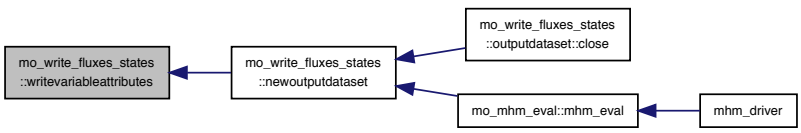
Date

June 2015

References mo_common_constants::nodata_dp.

Referenced by newoutputdataset().

Here is the caller graph for this function:



16.103.2.10 writevariabletimestep()

```
subroutine mo_write_fluxes_states::writevariabletimestep (
             class(outputvariable), intent(inout) self,
             integer(i4), intent(in) timestep )  [private]
```

Write timestep to file.

Write the content of the derived types's component 'data' to file, average if necessary

Parameters

in,out	*class(OutputVariable) :: self*	
in	*integer(i4) :: timestep*	-> index along the time dimension of the netcdf variable

Authors

David Schafer

Date

June 2015

References mo_common_constants::nodata_dp.

16.104 mo_xor4096 Module Reference

Data Types

- interface get_timeseed
- interface xor4096
- interface xor4096g

Functions/Subroutines

- subroutine get_timeseed_i4_0d (seed)
- subroutine get_timeseed_i4_1d (seed)
- subroutine get_timeseed_i8_0d (seed)
- subroutine get_timeseed_i8_1d (seed)
- subroutine xor4096s_0d (seed, SingleIntegerRN, save_state)
- subroutine xor4096s_1d (seed, SingleIntegerRN, save_state)
- subroutine xor4096f_0d (seed, SingleRealRN, save_state)
- subroutine xor4096f_1d (seed, SingleRealRN, save_state)
- subroutine xor4096l_0d (seed, DoubleIntegerRN, save_state)
- subroutine xor4096l_1d (seed, DoubleIntegerRN, save_state)
- subroutine xor4096d_0d (seed, DoubleRealRN, save_state)
- subroutine xor4096d_1d (seed, DoubleRealRN, save_state)
- subroutine xor4096gf_0d (seed, SingleRealRN, save_state)
- subroutine xor4096gf_1d (seed, SingleRealRN, save_state)
- subroutine xor4096gd_0d (seed, DoubleRealRN, save_state)
- subroutine xor4096gd_1d (seed, DoubleRealRN, save_state)

Variables

- integer(i4), parameter, public n_save_state = 132_i4

  Dimension of vector saving the state of a stream.

16.104.1 Function/Subroutine Documentation

16.104.1.1 get_timeseed_i4_0d()

```
subroutine mo_xor4096::get_timeseed_i4_0d (
            integer(i4), intent(inout) seed )  [private]
```

16.104.1.2 get_timeseed_i4_1d()

```
subroutine mo_xor4096::get_timeseed_i4_1d (
            integer(i4), dimension(:), intent(inout) seed )  [private]
```

16.104.1.3 get_timeseed_i8_0d()

```
subroutine mo_xor4096::get_timeseed_i8_0d (
            integer(i8), intent(inout) seed )  [private]
```

References mo_kind::i8.

16.104.1.4 get_timeseed_i8_1d()

```
subroutine mo_xor4096::get_timeseed_i8_1d (
            integer(i8), dimension(:), intent(inout) seed )  [private]
```

References mo_kind::i8.

16.104.1.5 xor4096d_0d()

```
subroutine mo_xor4096::xor4096d_0d (
            integer(i8), intent(in) seed,
            real(dp), intent(out) DoubleRealRN,
            integer(i8), dimension(n_save_state), intent(inout), optional save_state )  [private]
```

References n_save_state.

16.104.1.6 xor4096d_1d()

```
subroutine mo_xor4096::xor4096d_1d (
            integer(i8), dimension(:), intent(in) seed,
            real(dp), dimension(size(seed, 1)), intent(out) DoubleRealRN,
```

```
            integer(i8), dimension(size(seed, 1), n_save_state), intent(inout), optional
save_state )  [private]
```

References n_save_state.

16.104.1.7 xor4096f_0d()

```
subroutine mo_xor4096::xor4096f_0d (
            integer(i4), intent(in) seed,
            real(sp), intent(out) SingleRealRN,
            integer(i4), dimension(n_save_state), intent(inout), optional save_state )  [private]
```

References n_save_state.

16.104.1.8 xor4096f_1d()

```
subroutine mo_xor4096::xor4096f_1d (
            integer(i4), dimension(:), intent(in) seed,
            real(sp), dimension(size(seed)), intent(out) SingleRealRN,
            integer(i4), dimension(size(seed, 1), n_save_state), intent(inout), optional
save_state )  [private]
```

References n_save_state.

16.104.1.9 xor4096gd_0d()

```
subroutine mo_xor4096::xor4096gd_0d (
            integer(i8), intent(in) seed,
            real(dp), intent(out) DoubleRealRN,
            integer(i8), dimension(n_save_state), intent(inout), optional save_state )  [private]
```

References n_save_state.

16.104.1.10 xor4096gd_1d()

```
subroutine mo_xor4096::xor4096gd_1d (
            integer(i8), dimension(:), intent(in) seed,
            real(dp), dimension(size(seed)), intent(out) DoubleRealRN,
            integer(i8), dimension(size(seed), n_save_state), intent(inout), optional save_↩
state )  [private]
```

References n_save_state.

16.104.1.11 xor4096gf_0d()

```
subroutine mo_xor4096::xor4096gf_0d (
            integer(i4), intent(in) seed,
            real(sp), intent(out) SingleRealRN,
            integer(i4), dimension(n_save_state), intent(inout), optional save_state )  [private]
```

References n_save_state.

16.104.1.12 xor4096gf_1d()

```
subroutine mo_xor4096::xor4096gf_1d (
            integer(i4), dimension(:), intent(in) seed,
            real(sp), dimension(size(seed)), intent(out) SingleRealRN,
            integer(i4), dimension(size(seed), n_save_state), intent(inout), optional save_↩
state )  [private]
```

References n_save_state.

16.104.1.13 xor4096l_0d()

```
subroutine mo_xor4096::xor4096l_0d (
            integer(i8), intent(in) seed,
            integer(i8), intent(out) DoubleIntegerRN,
            integer(i8), dimension(n_save_state), intent(inout), optional save_state )  [private]
```

References n_save_state.

16.104.1.14 xor4096l_1d()

```
subroutine mo_xor4096::xor4096l_1d (
            integer(i8), dimension(:), intent(in) seed,
            integer(i8), dimension(size(seed, 1)), intent(out) DoubleIntegerRN,
            integer(i8), dimension(size(seed, 1), n_save_state), intent(inout), optional
save_state )  [private]
```

References n_save_state.

16.104.1.15 xor4096s_0d()

```
subroutine mo_xor4096::xor4096s_0d (
            integer(i4), intent(in) seed,
            integer(i4), intent(out) SingleIntegerRN,
            integer(i4), dimension(n_save_state), intent(inout), optional save_state )  [private]
```

References n_save_state.

16.104.1.16 xor4096s_1d()

```
subroutine mo_xor4096::xor4096s_1d (
            integer(i4), dimension(:), intent(in) seed,
            integer(i4), dimension(size(seed, 1)), intent(out) SingleIntegerRN,
            integer(i4), dimension(size(seed, 1), n_save_state), intent(inout), optional
save_state )  [private]
```

References n_save_state.

16.104.2 Variable Documentation

16.104.2.1 n_save_state

```
integer(i4), parameter, public mo_xor4096::n_save_state = 132_i4
```

Dimension of vector saving the state of a stream.

Referenced by mo_sce::cce(), mo_sce::getpnt(), mo_mcmc::mcmc_dp(), mo_mcmc::mcmc_stddev_dp(), mo↩_sce::sce(), xor4096d_0d(), xor4096d_1d(), xor4096f_0d(), xor4096f_1d(), xor4096gd_0d(), xor4096gd_1d(), xor4096gf_0d(), xor4096gf_1d(), xor4096l_0d(), xor4096l_1d(), xor4096s_0d(), and xor4096s_1d().

Chapter 17

Data Type Documentation

17.1 mo_moment::absdev Interface Reference

Public Member Functions

- real(sp) function absdev_sp (dat, mask)
- real(dp) function absdev_dp (dat, mask)

17.1.1 Member Function/Subroutine Documentation

17.1.1.1 absdev_dp()

```
real(dp) function mo_moment::absdev::absdev_dp (
             real(dp), dimension(:), intent(in) dat,
             logical, dimension(:), intent(in), optional mask )
```

17.1.1.2 absdev_sp()

```
real(sp) function mo_moment::absdev::absdev_sp (
             real(sp), dimension(:), intent(in) dat,
             logical, dimension(:), intent(in), optional mask )
```

The documentation for this interface was generated from the following file:

- mo_moment.f90

17.2 mo_anneal::anneal Interface Reference

anneal

Public Member Functions

- real(dp) function, dimension(size(para, 1)) anneal_dp (eval, cost, para, prange, prange_func, temp, Dt, nl↩ TERmax, Len, nST, eps, acc, seeds, printflag, maskpara, weight, changeParaMode, reflectionFlag, pertub↩ FlexFlag, maxit, undef_funcval, tmp_file, funcbest, history)

17.2.1 Detailed Description

anneal

Optimizes a user provided cost function using the Simulated Annealing strategy.

Parameters

in	*INTERFACE :: cost_dp*	interface calculating the cost function at a given point
in	*REAL(DP), DIMENSION(:) :: para*	initial parameter set
in	*REAL(DP), DIMENSION(size(para),2), optional :: prange*	lower and upper bound per parameter
in	*INTERFACE, optional :: prange_func*	interface calculating the feasible range for a parameter at a certain point, if ranges are variable
in	*REAL(DP), optional :: temp*	initial temperature DEFAULT: Get_Temperature
in	*REAL(DP), optional :: DT*	geometrical decreement of temperature 0.7<DT<0.999 DEFAULT: 0.9_dp
in	*INTEGER(I4), optional :: nITERmax*	maximal number of iterations will be increased by 10% if stopping criteria of acceptance ratio or epsilon decreement of cost function is not fullfilled DEFAULT: 1000_i4
in	*INTEGER(I4), optional :: LEN*	Length of Markov Chain DEFAULT: MAX(250_i4, size(para,1))
in	*INTEGER(I4), optional :: nST*	Number of consecutive LEN steps DEFAULT: 5_i4
in	*REAL(DP), optional :: eps*	Stopping criteria of epsilon decreement of cost function DEFAULT: 0.0001_dp
in	*REAL(DP), optional :: acc*	Stopping criteria for Acceptance Ratio acc <= 0.1_dp DEFAULT: 0.1_dp
in	*INTEGER(I4/I8), DIMENSION(3), optional :: seeds*	Seeds of random numbers used for random parameter set generation DEFAULT: dependent on current time
in	*LOGICAL, optional :: printflag*	If .true. detailed command line output is written DEFAULT: .false.
in	*LOGICAL, DIMENSION(size(para)), optional :: maskpara*	maskpara(i) = .true. –> parameter is optimized maskpara(i) = .false. –> parameter is discarded from optimiztaion DEFAULT: .true.
in	*REAL(DP), DIMENSION(size(para)), optional :: weight*	vector of weights per parameter: gives the frequency of parameter to be chosen for optimization (will be scaled to a CDF internally) eg. [1,2,1] –> parameter 2 is chosen twice as often as parameter 1 and 2 DEFAULT: weight = 1.0_dp
in	*INTEGER(I4), optional :: changeParaMode*	which and how many param.are changed in one step 1 = one parameter 2 = all parameter 3 = neighborhood parameter DEFAULT: 1_i4

Parameters

in	*LOGICAL, optional :: reflectionFlag*	if new parameter values are Gaussian distributed and reflected (.true.) or uniform in range (.false.) DEFAULT: .false.
in	*LOGICAL, optional :: pertubFlexFlag*	if pertubation of Gaussian distributed parameter values is constant at 0.2 (.false.) or depends on dR (.true.) DEFAULT: .true.
in	*LOGICAL, optional :: maxit*	maximizing (.true.) or minimizing (.false.) a function DEFAULT: .false. (minimization)
in	*REAL(DP), optional :: undef_funcval*	objective function value defining invalid model output, e.g. -999.0_dp
in	*character(len=*) , optional :: tmp_file*	file with temporal output
out	*REAL(DP), optional :: funcbest*	minimized value of cost function
out	*"REAL(DP),DIMENSION(,:),ALLOCATABLE,optional*	:: history" returns a vector of achieved objective

Returns

real(dp) :: parabest(size(para)) — Parameter set minimizing the cost function.

Note

Either fixed parameter range (prange) OR flexible parameter range (function interface prange_func) has to be given in calling sequence.
Only double precision version available.
If single precision is needed not only DP has to be replaced by SP but also I8 of save_state (random number variables) has to be replaced by I4.
ParaChangeMode $>$ 1 is not applied in GetTemperature. For Temperature estimation always only one single parameter is changed (ParaChangeMode=1) which should give theoretically always the best estimate.
Cost and prange_func are user defined functions. See interface definition.

17.2.2 Member Function/Subroutine Documentation

17.2.2.1 anneal_dp()

```
real(dp) function, dimension(size(para, 1)) mo_anneal::anneal::anneal_dp (
            procedure(eval_interface), intent(in), pointer eval,
            procedure(objective_interface), intent(in), pointer cost,
            real(dp), dimension(:), intent(in) para,
            real(dp), dimension(size(para, 1), 2), intent(in), optional prange,
            optional prange_func,
            real(dp), intent(in), optional temp,
            real(dp), intent(in), optional Dt,
            integer(i4), intent(in), optional nITERmax,
            integer(i4), intent(in), optional Len,
            integer(i4), intent(in), optional nST,
            real(dp), intent(in), optional eps,
            real(dp), intent(in), optional acc,
            integer(i8), dimension(3), intent(in), optional seeds,
```

```
logical, intent(in), optional printflag,
logical, dimension(size(para, 1)), intent(in), optional maskpara,
real(dp), dimension(size(para, 1)), intent(in), optional weight,
integer(i4), intent(in), optional changeParaMode,
logical, intent(in), optional reflectionFlag,
logical, intent(in), optional pertubFlexFlag,
logical, intent(in), optional maxit,
real(dp), intent(in), optional undef_funcval,
character(len = *), intent(in), optional tmp_file,
real(dp), intent(out), optional funcbest,
real(dp), dimension(:, :), intent(out), optional, allocatable history )
```

The documentation for this interface was generated from the following file:

- mo_anneal.f90

17.3 mo_append::append Interface Reference

Append (rows) scalars, vectors, and matrixes onto existing array.

Public Member Functions

- subroutine append_i4_v_s (vec1, sca2)
- subroutine append_i4_v_v (vec1, vec2)
- subroutine append_i4_m_m (mat1, mat2, fill_value)
- subroutine append_i8_v_s (vec1, sca2)
- subroutine append_i8_v_v (vec1, vec2)
- subroutine append_i8_m_m (mat1, mat2, fill_value)
- subroutine append_i8_3d (mat1, mat2, fill_value)
- subroutine append_sp_v_s (vec1, sca2)
- subroutine append_sp_v_v (vec1, vec2)
- subroutine append_sp_m_m (mat1, mat2, fill_value)
- subroutine append_sp_3d (mat1, mat2, fill_value)
- subroutine append_dp_v_s (vec1, sca2)
- subroutine append_dp_v_v (vec1, vec2)
- subroutine append_dp_m_m (mat1, mat2, fill_value)
- subroutine append_dp_3d (mat1, mat2, fill_value)
- subroutine append_char_v_s (vec1, sca2)
- subroutine append_char_v_v (vec1, vec2)
- subroutine append_char_m_m (mat1, mat2, fill_value)
- subroutine append_char_3d (mat1, mat2, fill_value)
- subroutine append_lgt_v_s (vec1, sca2)
- subroutine append_lgt_v_v (vec1, vec2)
- subroutine append_lgt_m_m (mat1, mat2, fill_value)
- subroutine append_lgt_3d (mat1, mat2, fill_value)

17.3.1 Detailed Description

Append (rows) scalars, vectors, and matrixes onto existing array.

Appends one input to the rows of another, i.e. append on the first dimension.
The input might be a scalar, a vector or a matrix.
Possibilities are:
(1) append scalar to vector

(2) append vector to vector
(3) append matrix to matrix

Parameters

in	*input2*	values to append. Can be INTEGER(I4/I8), REAL(SP/DP), or CHARACTER(len=∗) and also scalar, DIMENSION(:), or DIMENSION(:,:) Also includes 3d version, where the append is always done along the first dimension. If not scalar then the columns have to agree with input1.
in,out	*allocatable :: input1*	array to be appended to. Can be INTEGER(I4/I8), REAL(SP/DP), or CHARACTER(len=∗). Must be DIMENSION(:) or DIMENSION(:,:), and allocatable. If input2 is not scalar then it must be size(input1,2) = size(input2,2).

Author

Juliane Mai

Date

Aug 2012

17.3.2 Member Function/Subroutine Documentation

17.3.2.1 append_char_3d()

```
subroutine mo_append::append::append_char_3d (
             character(len = *), dimension(:, :, :), intent(inout), allocatable mat1,
             character(len = *), dimension(:, :, :), intent(in) mat2,
             character(len = *), intent(in), optional fill_value )
```

17.3.2.2 append_char_m_m()

```
subroutine mo_append::append::append_char_m_m (
             character(len = *), dimension(:, :), intent(inout), allocatable mat1,
             character(len = *), dimension(:, :), intent(in) mat2,
             character(len = *), intent(in), optional fill_value )
```

17.3.2.3 append_char_v_s()

```
subroutine mo_append::append::append_char_v_s (
             character(len = *), dimension(:), intent(inout), allocatable vec1,
             character(len = *), intent(in) sca2 )
```

17.3.2.4 append_char_v_v()

```
subroutine mo_append::append::append_char_v_v (
             character(len = *), dimension(:), intent(inout), allocatable vec1,
             character(len = *), dimension(:), intent(in) vec2 )
```

17.3.2.5 append_dp_3d()

```
subroutine mo_append::append::append_dp_3d (
             real(dp), dimension(:, :, :), intent(inout), allocatable mat1,
             real(dp), dimension(:, :, :), intent(in) mat2,
             real(dp), intent(in), optional fill_value )
```

17.3.2.6 append_dp_m_m()

```
subroutine mo_append::append::append_dp_m_m (
             real(dp), dimension(:, :), intent(inout), allocatable mat1,
             real(dp), dimension(:, :), intent(in) mat2,
             real(dp), intent(in), optional fill_value )
```

17.3.2.7 append_dp_v_s()

```
subroutine mo_append::append::append_dp_v_s (
             real(dp), dimension(:), intent(inout), allocatable vec1,
             real(dp), intent(in) sca2 )
```

17.3.2.8 append_dp_v_v()

```
subroutine mo_append::append::append_dp_v_v (
             real(dp), dimension(:), intent(inout), allocatable vec1,
             real(dp), dimension(:), intent(in) vec2 )
```

17.3.2.9 append_i4_m_m()

```
subroutine mo_append::append::append_i4_m_m (
             integer(i4), dimension(:, :), intent(inout), allocatable mat1,
             integer(i4), dimension(:, :), intent(in) mat2,
             integer(i4), intent(in), optional fill_value )
```

17.3.2.10 append_i4_v_s()

```
subroutine mo_append::append::append_i4_v_s (
             integer(i4), dimension(:), intent(inout), allocatable vec1,
             integer(i4), intent(in) sca2 )
```

17.3.2.11 append_i4_v_v()

```
subroutine mo_append::append::append_i4_v_v (
             integer(i4), dimension(:), intent(inout), allocatable vec1,
             integer(i4), dimension(:), intent(in) vec2 )
```

17.3.2.12 append_i8_3d()

```
subroutine mo_append::append::append_i8_3d (
             integer(i8), dimension(:, :, :), intent(inout), allocatable mat1,
             integer(i8), dimension(:, :, :), intent(in) mat2,
             integer(i8), intent(in), optional fill_value )
```

17.3.2.13 append_i8_m_m()

```
subroutine mo_append::append::append_i8_m_m (
             integer(i8), dimension(:, :), intent(inout), allocatable mat1,
             integer(i8), dimension(:, :), intent(in) mat2,
             integer(i8), intent(in), optional fill_value )
```

17.3.2.14 append_i8_v_s()

```
subroutine mo_append::append::append_i8_v_s (
             integer(i8), dimension(:), intent(inout), allocatable vec1,
             integer(i8), intent(in) sca2 )
```

17.3.2.15 append_i8_v_v()

```
subroutine mo_append::append::append_i8_v_v (
             integer(i8), dimension(:), intent(inout), allocatable vec1,
             integer(i8), dimension(:), intent(in) vec2 )
```

17.3.2.16 append_lgt_3d()

```
subroutine mo_append::append::append_lgt_3d (
             logical, dimension(:, :, :), intent(inout), allocatable mat1,
             logical, dimension(:, :, :), intent(in) mat2,
             logical, intent(in), optional fill_value )
```

17.3.2.17 append_lgt_m_m()

```
subroutine mo_append::append::append_lgt_m_m (
             logical, dimension(:, :), intent(inout), allocatable mat1,
             logical, dimension(:, :), intent(in) mat2,
             logical, intent(in), optional fill_value )
```

17.3.2.18 append_lgt_v_s()

```
subroutine mo_append::append::append_lgt_v_s (
             logical, dimension(:), intent(inout), allocatable vec1,
             logical, intent(in) sca2 )
```

17.3.2.19 append_lgt_v_v()

```
subroutine mo_append::append::append_lgt_v_v (
             logical, dimension(:), intent(inout), allocatable vec1,
             logical, dimension(:), intent(in) vec2 )
```

17.3.2.20 append_sp_3d()

```
subroutine mo_append::append::append_sp_3d (
             real(sp), dimension(:, :, :), intent(inout), allocatable mat1,
             real(sp), dimension(:, :, :), intent(in) mat2,
             real(sp), intent(in), optional fill_value )
```

17.3.2.21 append_sp_m_m()

```
subroutine mo_append::append::append_sp_m_m (
             real(sp), dimension(:, :), intent(inout), allocatable mat1,
             real(sp), dimension(:, :), intent(in) mat2,
             real(sp), intent(in), optional fill_value )
```

17.3.2.22 append_sp_v_s()

```
subroutine mo_append::append::append_sp_v_s (
             real(sp), dimension(:), intent(inout), allocatable vec1,
             real(sp), intent(in) sca2 )
```

17.3.2.23 append_sp_v_v()

```
subroutine mo_append::append::append_sp_v_v (
             real(sp), dimension(:), intent(inout), allocatable vec1,
             real(sp), dimension(:), intent(in) vec2 )
```

The documentation for this interface was generated from the following file:

- mo_append.f90

17.4 mo_corr::arth Interface Reference

Private Member Functions

- real(sp) function, dimension(n) arth_sp (first, increment, n)
- real(dp) function, dimension(n) arth_dp (first, increment, n)
- integer(i4) function, dimension(n) arth_i4 (first, increment, n)

17.4.1 Member Function/Subroutine Documentation

17.4.1.1 arth_dp()

```
real(dp) function, dimension(n) mo_corr::arth::arth_dp (
             real(dp), intent(in) first,
             real(dp), intent(in) increment,
             integer(i4), intent(in) n )  [private]
```

17.4.1.2 arth_i4()

```
integer(i4) function, dimension(n) mo_corr::arth::arth_i4 (
             integer(i4), intent(in) first,
             integer(i4), intent(in) increment,
             integer(i4), intent(in) n )  [private]
```

17.4.1.3 arth_sp()

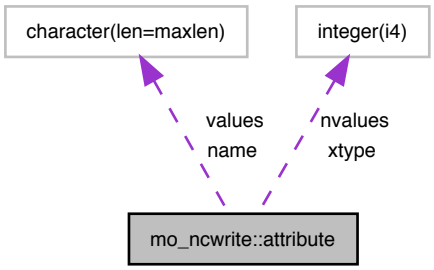
```
real(sp) function, dimension(n) mo_corr::arth::arth_sp (
             real(sp), intent(in) first,
             real(sp), intent(in) increment,
             integer(i4), intent(in) n )  [private]
```

The documentation for this interface was generated from the following file:

- mo_corr.f90

17.5 mo_ncwrite::attribute Type Reference

Collaboration diagram for mo_ncwrite::attribute:



Public Attributes

- character(len=maxlen) name
- integer(i4) xtype
- integer(i4) nvalues
- character(len=maxlen) values

17.5.1 Member Data Documentation

17.5.1.1 name

```
character (len = maxlen) mo_ncwrite::attribute::name
```

17.5.1.2 nvalues

```
integer(i4) mo_ncwrite::attribute::nvalues
```

17.5.1.3 values

```
character (len = maxlen) mo_ncwrite::attribute::values
```

17.5.1.4 xtype

```
integer(i4) mo_ncwrite::attribute::xtype
```

The documentation for this type was generated from the following file:

- mo_ncwrite.f90

17.6 mo_corr::autocoeffk Interface Reference

Public Member Functions

- real(sp) function autocoeffk_sp (x, k, mask)
- real(dp) function autocoeffk_dp (x, k, mask)
- real(dp) function, dimension(size(k)) autocoeffk_1d_dp (x, k, mask)
- real(sp) function, dimension(size(k)) autocoeffk_1d_sp (x, k, mask)

17.6.1 Member Function/Subroutine Documentation

17.6.1.1 autocoeffk_1d_dp()

```
real(dp) function, dimension(size(k)) mo_corr::autocoeffk::autocoeffk_1d_dp (
            real(dp), dimension(:), intent(in) x,
            integer(i4), dimension(:), intent(in) k,
            logical, dimension(:), intent(in), optional mask )
```

17.6.1.2 autocoeffk_1d_sp()

```
real(sp) function, dimension(size(k)) mo_corr::autocoeffk::autocoeffk_1d_sp (
            real(sp), dimension(:), intent(in) x,
            integer(i4), dimension(:), intent(in) k,
            logical, dimension(:), intent(in), optional mask )
```

17.6.1.3 autocoeffk_dp()

```
real(dp) function mo_corr::autocoeffk::autocoeffk_dp (
            real(dp), dimension(:), intent(in) x,
            integer(i4), intent(in) k,
            logical, dimension(:), intent(in), optional mask )
```

17.6.1.4 autocoeffk_sp()

```
real(sp) function mo_corr::autocoeffk::autocoeffk_sp (
            real(sp), dimension(:), intent(in) x,
            integer(i4), intent(in) k,
            logical, dimension(:), intent(in), optional mask )
```

The documentation for this interface was generated from the following file:

- mo_corr.f90

17.7 mo_corr::autocorr Interface Reference

Public Member Functions

- real(sp) function autocorr_sp (x, k, mask)
- real(dp) function autocorr_dp (x, k, mask)
- real(sp) function, dimension(size(k)) autocorr_1d_sp (x, k, mask)
- real(dp) function, dimension(size(k)) autocorr_1d_dp (x, k, mask)

17.7.1 Member Function/Subroutine Documentation

17.7.1.1 autocorr_1d_dp()

```
real(dp) function, dimension(size(k)) mo_corr::autocorr::autocorr_1d_dp (
            real(dp), dimension(:), intent(in) x,
            integer(i4), dimension(:), intent(in) k,
            logical, dimension(:), intent(in), optional mask )
```

17.7.1.2 autocorr_1d_sp()

```
real(sp) function, dimension(size(k)) mo_corr::autocorr::autocorr_1d_sp (
            real(sp), dimension(:), intent(in) x,
            integer(i4), dimension(:), intent(in) k,
            logical, dimension(:), intent(in), optional mask )
```

17.7.1.3 autocorr_dp()

```
real(dp) function mo_corr::autocorr::autocorr_dp (
            real(dp), dimension(:), intent(in) x,
            integer(i4), intent(in) k,
            logical, dimension(:), intent(in), optional mask )
```

17.7.1.4 autocorr_sp()

```
real(sp) function mo_corr::autocorr::autocorr_sp (
            real(sp), dimension(:), intent(in) x,
            integer(i4), intent(in) k,
            logical, dimension(:), intent(in), optional mask )
```

The documentation for this interface was generated from the following file:

- mo_corr.f90

17.8 mo_moment::average Interface Reference

Public Member Functions

- real(sp) function average_sp (dat, mask)
- real(dp) function average_dp (dat, mask)

17.8.1 Member Function/Subroutine Documentation

17.8.1.1 average_dp()

```
real(dp) function mo_moment::average::average_dp (
            real(dp), dimension(:), intent(in) dat,
            logical, dimension(:), intent(in), optional mask )
```

17.8.1.2 average_sp()

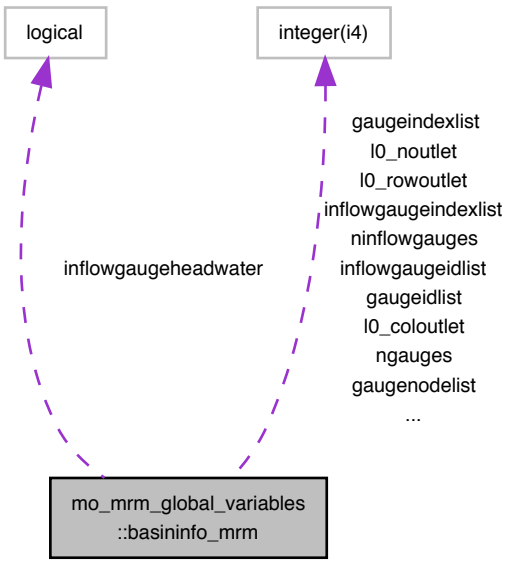
```
real(sp) function mo_moment::average::average_sp (
            real(sp), dimension(:), intent(in) dat,
            logical, dimension(:), intent(in), optional mask )
```

The documentation for this interface was generated from the following file:

- mo_moment.f90

17.9 mo_mrm_global_variables::basininfo_mrm Type Reference

Collaboration diagram for mo_mrm_global_variables::basininfo_mrm:



Public Attributes

- integer(i4) ngauges
- integer(i4), dimension(:), allocatable gaugeidlist
- integer(i4), dimension(:), allocatable gaugeindexlist
- integer(i4), dimension(:), allocatable gaugenodelist
- integer(i4) ninflowgauges
- integer(i4), dimension(:), allocatable inflowgaugeidlist
- integer(i4), dimension(:), allocatable inflowgaugeindexlist
- integer(i4), dimension(:), allocatable inflowgaugenodelist
- logical, dimension(:), allocatable inflowgaugeheadwater
- integer(i4) l0_noutlet
- integer(i4), dimension(:), allocatable l0_rowoutlet
- integer(i4), dimension(:), allocatable l0_coloutlet

17.9.1 Member Data Documentation

17.9.1.1 gaugeidlist

```
integer(i4), dimension(:), allocatable mo_mrm_global_variables::basininfo_mrm::gaugeidlist
```

17.9.1.2 gaugeindexlist

```
integer(i4), dimension(:), allocatable mo_mrm_global_variables::basininfo_mrm::gaugeindexlist
```

17.9.1.3 gaugenodelist

```
integer(i4), dimension(:), allocatable mo_mrm_global_variables::basininfo_mrm::gaugenodelist
```

17.9.1.4 inflowgaugeheadwater

```
logical, dimension(:), allocatable mo_mrm_global_variables::basininfo_mrm::inflowgaugeheadwater
```

17.9.1.5 inflowgaugeidlist

```
integer(i4), dimension(:), allocatable mo_mrm_global_variables::basininfo_mrm::inflowgaugeidlist
```

17.9.1.6 inflowgaugeindexlist

```
integer(i4), dimension(:), allocatable mo_mrm_global_variables::basininfo_mrm::inflowgaugeindexlist
```

17.9.1.7 inflowgaugenodelist

```
integer(i4), dimension(:), allocatable mo_mrm_global_variables::basininfo_mrm::inflowgaugenodelist
```

17.9.1.8 l0_coloutlet

```
integer(i4), dimension(:), allocatable mo_mrm_global_variables::basininfo_mrm::l0_coloutlet
```

17.9.1.9 l0_noutlet

```
integer(i4) mo_mrm_global_variables::basininfo_mrm::l0_noutlet
```

17.9.1.10 l0_rowoutlet

```
integer(i4), dimension(:), allocatable mo_mrm_global_variables::basininfo_mrm::l0_rowoutlet
```

17.9.1.11 ngauges

```
integer(i4) mo_mrm_global_variables::basininfo_mrm::ngauges
```

17.9.1.12 ninflowgauges

```
integer(i4) mo_mrm_global_variables::basininfo_mrm::ninflowgauges
```

The documentation for this type was generated from the following file:

- mo_mrm_global_variables.f90

17.10 mo_errormeasures::bias Interface Reference

Public Member Functions

- real(sp) function bias_sp_1d (x, y, mask)
- real(dp) function bias_dp_1d (x, y, mask)
- real(sp) function bias_sp_2d (x, y, mask)
- real(dp) function bias_dp_2d (x, y, mask)
- real(sp) function bias_sp_3d (x, y, mask)
- real(dp) function bias_dp_3d (x, y, mask)

17.10.1 Member Function/Subroutine Documentation

17.10.1.1 bias_dp_1d()

```
real(dp) function mo_errormeasures::bias::bias_dp_1d (
             real(dp), dimension(:), intent(in) x,
             real(dp), dimension(:), intent(in) y,
             logical, dimension(:), intent(in), optional mask )
```

17.10.1.2 bias_dp_2d()

```
real(dp) function mo_errormeasures::bias::bias_dp_2d (
             real(dp), dimension(:, :), intent(in) x,
             real(dp), dimension(:, :), intent(in) y,
             logical, dimension(:, :), intent(in), optional mask )
```

17.10.1.3 bias_dp_3d()

```
real(dp) function mo_errormeasures::bias::bias_dp_3d (
             real(dp), dimension(:, :, :), intent(in) x,
             real(dp), dimension(:, :, :), intent(in) y,
             logical, dimension(:, :, :), intent(in), optional mask )
```

17.10.1.4 bias_sp_1d()

```
real(sp) function mo_errormeasures::bias::bias_sp_1d (
             real(sp), dimension(:), intent(in) x,
             real(sp), dimension(:), intent(in) y,
             logical, dimension(:), intent(in), optional mask )
```

17.10.1.5 bias_sp_2d()

```
real(sp) function mo_errormeasures::bias::bias_sp_2d (
             real(sp), dimension(:, :), intent(in) x,
             real(sp), dimension(:, :), intent(in) y,
             logical, dimension(:, :), intent(in), optional mask )
```

17.10.1.6 bias_sp_3d()

```
real(sp) function mo_errormeasures::bias::bias_sp_3d (
             real(sp), dimension(:, :, :), intent(in) x,
             real(sp), dimension(:, :, :), intent(in) y,
             logical, dimension(:, :, :), intent(in), optional mask )
```

The documentation for this interface was generated from the following file:

- mo_errormeasures.f90

17.11 mo_moment::central_moment Interface Reference

Public Member Functions

- real(sp) function central_moment_sp (x, r, mask)
- real(dp) function central_moment_dp (x, r, mask)

17.11.1 Member Function/Subroutine Documentation

17.11.1.1 central_moment_dp()

```
real(dp) function mo_moment::central_moment::central_moment_dp (
             real(dp), dimension(:), intent(in) x,
             integer(i4), intent(in) r,
             logical, dimension(:), intent(in), optional mask )
```

17.11.1.2 central_moment_sp()

```
real(sp) function mo_moment::central_moment::central_moment_sp (
             real(sp), dimension(:), intent(in) x,
             integer(i4), intent(in) r,
             logical, dimension(:), intent(in), optional mask )
```

The documentation for this interface was generated from the following file:

- mo_moment.f90

17.12 mo_moment::central_moment_var Interface Reference

Public Member Functions

- real(sp) function central_moment_var_sp (x, r, mask)
- real(dp) function central_moment_var_dp (x, r, mask)

17.12.1 Member Function/Subroutine Documentation

17.12.1.1 central_moment_var_dp()

```
real(dp) function mo_moment::central_moment_var::central_moment_var_dp (
             real(dp), dimension(:), intent(in) x,
             integer(i4), intent(in) r,
             logical, dimension(:), intent(in), optional mask )
```

17.12.1.2 central_moment_var_sp()

```
real(sp) function mo_moment::central_moment_var::central_moment_var_sp (
            real(sp), dimension(:), intent(in) x,
            integer(i4), intent(in) r,
            logical, dimension(:), intent(in), optional mask )
```

The documentation for this interface was generated from the following file:

- mo_moment.f90

17.13 mo_standard_score::classified_standard_score Interface Reference

Calculates the classified standard score (e.g. classes are months).

Public Member Functions

- real(sp) function, dimension(size(data, dim=1)) classified_standard_score_sp (data, classes, mask)
- real(dp) function, dimension(size(data, dim=1)) classified_standard_score_dp (data, classes, mask)

17.13.1 Detailed Description

Calculates the classified standard score (e.g. classes are months).

In statistics, the standard score is the (signed) number of standard deviations an observation or datum is above the mean. Thus, a positive standard score indicates a datum above the mean, while a negative standard score indicates a datum below the mean. It is a dimensionless quantity obtained by subtracting the population mean from an individual raw score and then dividing the difference by the population standard deviation. This conversion process is called standardizing or normalizing (however, "normalizing" can refer to many types of ratios).
Standard scores are also called z-values, z-scores, normal scores, and standardized variables; the use of "Z" is because the normal distribution is also known as the "Z distribution". They are most frequently used to compare a sample to a standard normal deviate, though they can be defined without assumptions of normality (Wikipedia, May 2015).
In this particular case the standard score is calculated for means and standard deviations derived from classes of the time series. Such classes could be for example months. Thus, the output would be a deseasonalized time series.

$$classified\_standard\_score = \frac{x_i - \mu_{c_{x_i}}}{\sigma_{c_{x_i}}}$$

where x_i is an element of class c_{x_i}. x is a population, $\mu_{c_{x_i}}$ is the mean of all members of a class c_{x_i} and $\sigma_{c_{x_i}}$ its standard deviation.

If an optinal mask is given, the calculations are over those locations that correspond to true values in the mask. data can be single or double precision. The result will have the same numerical precision.

Parameters

in	*integer, dimension(:) :: classes*	classes to categorize data (e.g. months)
in	*real(sp/dp), dimension(:) :: data*	data to calculate the standard score for
in	*logical, dimension(:), optional :: mask*	indication which cells to use for calculation If present, only those locations in mask having true values in mask are evaluated.

Returns

real(sp/dp) :: classified_standard_score — classified standard score (e.g. deseasonalized time series)

Author

Matthias Zink

Date

May 2015

17.13.2 Member Function/Subroutine Documentation

17.13.2.1 classified_standard_score_dp()

```
real(dp) function, dimension(size(data, dim = 1)) mo_standard_score::classified_standard_↩
score::classified_standard_score_dp (
            real(dp), dimension(:), intent(in) data,
            integer, dimension(:), intent(in) classes,
            logical, dimension(:), intent(in), optional mask )
```

17.13.2.2 classified_standard_score_sp()

```
real(sp) function, dimension(size(data, dim = 1)) mo_standard_score::classified_standard_↩
score::classified_standard_score_sp (
            real(sp), dimension(:), intent(in) data,
            integer, dimension(:), intent(in) classes,
            logical, dimension(:), intent(in), optional mask )
```

The documentation for this interface was generated from the following file:

- mo_standard_score.f90

17.14 mo_corr::corr Interface Reference

Public Member Functions

- real(sp) function, dimension(size(data1)) corr_sp (data1, data2, nadjust, nhigh, nwin)
- real(dp) function, dimension(size(data1)) corr_dp (data1, data2, nadjust, nhigh, nwin)

17.14.1 Member Function/Subroutine Documentation

17.14.1.1 corr_dp()

```
real(dp) function, dimension(size(data1)) mo_corr::corr::corr_dp (
            real(dp), dimension(:), intent(in) data1,
```

```
            real(dp), dimension(:), intent(in) data2,
            integer(i4), intent(out), optional nadjust,
            integer(i4), intent(in), optional nhigh,
            integer(i4), intent(in), optional nwin )
```

17.14.1.2 corr_sp()

```
real(sp) function, dimension(size(data1)) mo_corr::corr::corr_sp (
            real(sp), dimension(:), intent(in) data1,
            real(sp), dimension(:), intent(in) data2,
            integer(i4), intent(out), optional nadjust,
            integer(i4), intent(in), optional nhigh,
            integer(i4), intent(in), optional nwin )
```

The documentation for this interface was generated from the following file:

- mo_corr.f90

17.15 mo_moment::correlation Interface Reference

Public Member Functions

- real(sp) function correlation_sp (x, y, mask)
- real(dp) function correlation_dp (x, y, mask)

17.15.1 Member Function/Subroutine Documentation

17.15.1.1 correlation_dp()

```
real(dp) function mo_moment::correlation::correlation_dp (
            real(dp), dimension(:), intent(in) x,
            real(dp), dimension(:), intent(in) y,
            logical, dimension(:), intent(in), optional mask )
```

17.15.1.2 correlation_sp()

```
real(sp) function mo_moment::correlation::correlation_sp (
            real(sp), dimension(:), intent(in) x,
            real(sp), dimension(:), intent(in) y,
            logical, dimension(:), intent(in), optional mask )
```

The documentation for this interface was generated from the following file:

- mo_moment.f90

17.16 mo_moment::covariance Interface Reference

Public Member Functions

- real(sp) function covariance_sp (x, y, mask)
- real(dp) function covariance_dp (x, y, mask)

17.16.1 Member Function/Subroutine Documentation

17.16.1.1 covariance_dp()

```
real(dp) function mo_moment::covariance::covariance_dp (
             real(dp), dimension(:), intent(in) x,
             real(dp), dimension(:), intent(in) y,
             logical, dimension(:), intent(in), optional mask )
```

17.16.1.2 covariance_sp()

```
real(sp) function mo_moment::covariance::covariance_sp (
             real(sp), dimension(:), intent(in) x,
             real(sp), dimension(:), intent(in) y,
             logical, dimension(:), intent(in), optional mask )
```

The documentation for this interface was generated from the following file:

- mo_moment.f90

17.17 mo_corr::crosscoeffk Interface Reference

Public Member Functions

- real(sp) function crosscoeffk_sp (x, y, k, mask)
- real(dp) function crosscoeffk_dp (x, y, k, mask)

17.17.1 Member Function/Subroutine Documentation

17.17.1.1 crosscoeffk_dp()

```
real(dp) function mo_corr::crosscoeffk::crosscoeffk_dp (
             real(dp), dimension(:), intent(in) x,
             real(dp), dimension(:), intent(in) y,
             integer(i4), intent(in) k,
             logical, dimension(:), intent(in), optional mask )
```

17.17.1.2 crosscoeffk_sp()

```
real(sp) function mo_corr::crosscoeffk::crosscoeffk_sp (
             real(sp), dimension(:), intent(in) x,
             real(sp), dimension(:), intent(in) y,
             integer(i4), intent(in) k,
             logical, dimension(:), intent(in), optional mask )
```

The documentation for this interface was generated from the following file:

- mo_corr.f90

17.18 mo_corr::crosscorr Interface Reference

Public Member Functions

- real(sp) function crosscorr_sp (x, y, k, mask)
- real(dp) function crosscorr_dp (x, y, k, mask)

17.18.1 Member Function/Subroutine Documentation

17.18.1.1 crosscorr_dp()

```
real(dp) function mo_corr::crosscorr::crosscorr_dp (
             real(dp), dimension(:), intent(in) x,
             real(dp), dimension(:), intent(in) y,
             integer(i4), intent(in) k,
             logical, dimension(:), intent(in), optional mask )
```

17.18.1.2 crosscorr_sp()

```
real(sp) function mo_corr::crosscorr::crosscorr_sp (
             real(sp), dimension(:), intent(in) x,
             real(sp), dimension(:), intent(in) y,
             integer(i4), intent(in) k,
             logical, dimension(:), intent(in), optional mask )
```

The documentation for this interface was generated from the following file:

- mo_corr.f90

17.19 mo_orderpack::ctrper Interface Reference

Public Member Functions

- subroutine d_ctrper (XDONT, PCLS)
- subroutine r_ctrper (XDONT, PCLS)
- subroutine i_ctrper (XDONT, PCLS)

17.19.1 Member Function/Subroutine Documentation

17.19.1.1 d_ctrper()

```
subroutine mo_orderpack::ctrper::d_ctrper (
             real(kind = dp), dimension (:), intent(inout) XDONT,
             real(kind = dp), intent(in) PCLS )
```

17.19.1.2 i_ctrper()

```
subroutine mo_orderpack::ctrper::i_ctrper (
             integer(kind = i4), dimension (:), intent(inout) XDONT,
             real(kind = sp), intent(in) PCLS )
```

17.19.1.3 r_ctrper()

```
subroutine mo_orderpack::ctrper::r_ctrper (
             real(kind = sp), dimension (:), intent(inout) XDONT,
             real(kind = sp), intent(in) PCLS )
```

The documentation for this interface was generated from the following file:

- mo_orderpack.f90

17.20 mo_temporal_aggregation::day2mon_average Interface Reference

Day-to-month average (day2mon_average)

Public Member Functions

- subroutine day2mon_average_dp (daily_data, yearS, monthS, dayS, mon_avg, misval, rm_misval)

17.20.1 Detailed Description

Day-to-month average (day2mon_average)

converts daily time series to monthly

Parameters

in	*real(sp/dp) :: daily_data(:)*	array of daily time series
in	*integer(i4) :: year*	year of the starting time
in	*integer(i4) :: month*	month of the starting time
in	*integer(i4) :: day*	day of the starting time
in	*real(sp/dp) :: mon_average(:)*	array of monthly averaged values
in	*real(sp/dp) :: misval*	missing value definition
in	*logical :: rm_misval*	switch to exclude missing values

Note

Author

Oldrich Rakovec, Rohini Kumar

Date

Oct 2015

17.20.2 Member Function/Subroutine Documentation

17.20.2.1 day2mon_average_dp()

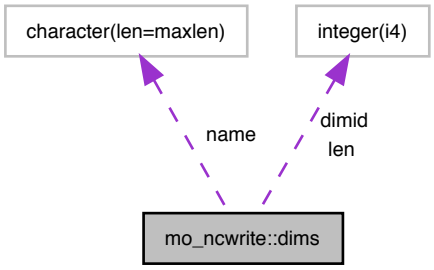
```
subroutine mo_temporal_aggregation::day2mon_average::day2mon_average_dp (
             real(dp), dimension(:), intent(in) daily_data,
             integer(i4), intent(in) yearS,
             integer(i4), intent(in) monthS,
             integer(i4), intent(in) dayS,
             real(dp), dimension(:), intent(inout), allocatable mon_avg,
             real(dp), intent(in), optional misval,
             logical, intent(in), optional rm_misval )
```

The documentation for this interface was generated from the following file:

- mo_temporal_aggregation.f90

17.21 mo_ncwrite::dims Type Reference

Collaboration diagram for mo_ncwrite::dims:



Public Attributes

- character(len=maxlen) name
- integer(i4) len
- integer(i4) dimid

17.21.1 Member Data Documentation

17.21.1.1 dimid

```
integer(i4) mo_ncwrite::dims::dimid
```

17.21.1.2 len

```
integer(i4) mo_ncwrite::dims::len
```

17.21.1.3 name

```
character (len = maxlen) mo_ncwrite::dims::name
```

The documentation for this type was generated from the following file:

- mo_ncwrite.f90

17.22 mo_ncwrite::dump_netcdf Interface Reference

Public Member Functions

- subroutine dump_netcdf_1d_sp (filename, arr, append, lfs, netcdf4, deflate_level)
- subroutine dump_netcdf_2d_sp (filename, arr, append, lfs, netcdf4, deflate_level)
- subroutine dump_netcdf_3d_sp (filename, arr, append, lfs, netcdf4, deflate_level)
- subroutine dump_netcdf_4d_sp (filename, arr, append, lfs, netcdf4, deflate_level)
- subroutine dump_netcdf_5d_sp (filename, arr, append, lfs, netcdf4, deflate_level)
- subroutine dump_netcdf_1d_dp (filename, arr, append, lfs, netcdf4, deflate_level)
- subroutine dump_netcdf_2d_dp (filename, arr, append, lfs, netcdf4, deflate_level)
- subroutine dump_netcdf_3d_dp (filename, arr, append, lfs, netcdf4, deflate_level)
- subroutine dump_netcdf_4d_dp (filename, arr, append, lfs, netcdf4, deflate_level)
- subroutine dump_netcdf_5d_dp (filename, arr, append, lfs, netcdf4, deflate_level)
- subroutine dump_netcdf_1d_i4 (filename, arr, append, lfs, netcdf4, deflate_level)
- subroutine dump_netcdf_2d_i4 (filename, arr, append, lfs, netcdf4, deflate_level)
- subroutine dump_netcdf_3d_i4 (filename, arr, append, lfs, netcdf4, deflate_level)
- subroutine dump_netcdf_4d_i4 (filename, arr, append, lfs, netcdf4, deflate_level)
- subroutine dump_netcdf_5d_i4 (filename, arr, append, lfs, netcdf4, deflate_level)

17.22.1 Member Function/Subroutine Documentation

17.22.1.1 dump_netcdf_1d_dp()

```
subroutine mo_ncwrite::dump_netcdf::dump_netcdf_1d_dp (
             character(len = *), intent(in) filename,
             real(dp), dimension(:), intent(in) arr,
             logical, intent(in), optional append,
             logical, intent(in), optional lfs,
             logical, intent(in), optional netcdf4,
             integer(i4), intent(in), optional deflate_level )
```

17.22.1.2 dump_netcdf_1d_i4()

```
subroutine mo_ncwrite::dump_netcdf::dump_netcdf_1d_i4 (
             character(len = *), intent(in) filename,
             integer(i4), dimension(:), intent(in) arr,
             logical, intent(in), optional append,
             logical, intent(in), optional lfs,
             logical, intent(in), optional netcdf4,
             integer(i4), intent(in), optional deflate_level )
```

17.22.1.3 dump_netcdf_1d_sp()

```
subroutine mo_ncwrite::dump_netcdf::dump_netcdf_1d_sp (
             character(len = *), intent(in) filename,
             real(sp), dimension(:), intent(in) arr,
             logical, intent(in), optional append,
             logical, intent(in), optional lfs,
             logical, intent(in), optional netcdf4,
             integer(i4), intent(in), optional deflate_level )
```

17.22.1.4 dump_netcdf_2d_dp()

```
subroutine mo_ncwrite::dump_netcdf::dump_netcdf_2d_dp (
             character(len = *), intent(in) filename,
             real(dp), dimension(:, :), intent(in) arr,
             logical, intent(in), optional append,
             logical, intent(in), optional lfs,
             logical, intent(in), optional netcdf4,
             integer(i4), intent(in), optional deflate_level )
```

17.22.1.5 dump_netcdf_2d_i4()

```
subroutine mo_ncwrite::dump_netcdf::dump_netcdf_2d_i4 (
             character(len = *), intent(in) filename,
             integer(i4), dimension(:, :), intent(in) arr,
             logical, intent(in), optional append,
             logical, intent(in), optional lfs,
             logical, intent(in), optional netcdf4,
             integer(i4), intent(in), optional deflate_level )
```

17.22.1.6 dump_netcdf_2d_sp()

```
subroutine mo_ncwrite::dump_netcdf::dump_netcdf_2d_sp (
             character(len = *), intent(in) filename,
             real(sp), dimension(:, :), intent(in) arr,
             logical, intent(in), optional append,
             logical, intent(in), optional lfs,
             logical, intent(in), optional netcdf4,
             integer(i4), intent(in), optional deflate_level )
```

17.22.1.7 dump_netcdf_3d_dp()

```
subroutine mo_ncwrite::dump_netcdf::dump_netcdf_3d_dp (
             character(len = *), intent(in) filename,
             real(dp), dimension(:, :, :), intent(in) arr,
             logical, intent(in), optional append,
             logical, intent(in), optional lfs,
             logical, intent(in), optional netcdf4,
             integer(i4), intent(in), optional deflate_level )
```

17.22.1.8 dump_netcdf_3d_i4()

```
subroutine mo_ncwrite::dump_netcdf::dump_netcdf_3d_i4 (
             character(len = *), intent(in) filename,
             integer(i4), dimension(:, :, :), intent(in) arr,
             logical, intent(in), optional append,
             logical, intent(in), optional lfs,
             logical, intent(in), optional netcdf4,
             integer(i4), intent(in), optional deflate_level )
```

17.22.1.9 dump_netcdf_3d_sp()

```
subroutine mo_ncwrite::dump_netcdf::dump_netcdf_3d_sp (
             character(len = *), intent(in) filename,
             real(sp), dimension(:, :, :), intent(in) arr,
             logical, intent(in), optional append,
             logical, intent(in), optional lfs,
             logical, intent(in), optional netcdf4,
             integer(i4), intent(in), optional deflate_level )
```

17.22.1.10 dump_netcdf_4d_dp()

```
subroutine mo_ncwrite::dump_netcdf::dump_netcdf_4d_dp (
             character(len = *), intent(in) filename,
             real(dp), dimension(:, :, :, :), intent(in) arr,
             logical, intent(in), optional append,
```

```
            logical, intent(in), optional lfs,
            logical, intent(in), optional netcdf4,
            integer(i4), intent(in), optional deflate_level )
```

17.22.1.11 dump_netcdf_4d_i4()

```
subroutine mo_ncwrite::dump_netcdf::dump_netcdf_4d_i4 (
            character(len = *), intent(in) filename,
            integer(i4), dimension(:, :, :, :), intent(in) arr,
            logical, intent(in), optional append,
            logical, intent(in), optional lfs,
            logical, intent(in), optional netcdf4,
            integer(i4), intent(in), optional deflate_level )
```

17.22.1.12 dump_netcdf_4d_sp()

```
subroutine mo_ncwrite::dump_netcdf::dump_netcdf_4d_sp (
            character(len = *), intent(in) filename,
            real(sp), dimension(:, :, :, :), intent(in) arr,
            logical, intent(in), optional append,
            logical, intent(in), optional lfs,
            logical, intent(in), optional netcdf4,
            integer(i4), intent(in), optional deflate_level )
```

17.22.1.13 dump_netcdf_5d_dp()

```
subroutine mo_ncwrite::dump_netcdf::dump_netcdf_5d_dp (
            character(len = *), intent(in) filename,
            real(dp), dimension(:, :, :, :, :), intent(in) arr,
            logical, intent(in), optional append,
            logical, intent(in), optional lfs,
            logical, intent(in), optional netcdf4,
            integer(i4), intent(in), optional deflate_level )
```

17.22.1.14 dump_netcdf_5d_i4()

```
subroutine mo_ncwrite::dump_netcdf::dump_netcdf_5d_i4 (
            character(len = *), intent(in) filename,
            integer(i4), dimension(:, :, :, :, :), intent(in) arr,
            logical, intent(in), optional append,
            logical, intent(in), optional lfs,
            logical, intent(in), optional netcdf4,
            integer(i4), intent(in), optional deflate_level )
```

17.22.1.15 dump_netcdf_5d_sp()

```
subroutine mo_ncwrite::dump_netcdf::dump_netcdf_5d_sp (
```

```
            character(len = *), intent(in) filename,
            real(sp), dimension(:, :, :, :, :), intent(in) arr,
            logical, intent(in), optional append,
            logical, intent(in), optional lfs,
            logical, intent(in), optional netcdf4,
            integer(i4), intent(in), optional deflate_level )
```

The documentation for this interface was generated from the following file:

- mo_ncwrite.f90

17.23 mo_utils::eq Interface Reference

Public Member Functions

- elemental pure logical function equal_sp (a, b)
- elemental pure logical function equal_dp (a, b)

17.23.1 Member Function/Subroutine Documentation

17.23.1.1 equal_dp()

```
elemental pure logical function mo_utils::eq::equal_dp (
            real(dp), intent(in) a,
            real(dp), intent(in) b )
```

17.23.1.2 equal_sp()

```
elemental pure logical function mo_utils::eq::equal_sp (
            real(sp), intent(in) a,
            real(sp), intent(in) b )
```

The documentation for this interface was generated from the following file:

- mo_utils.f90

17.24 mo_utils::equal Interface Reference

Comparison of real values.

Public Member Functions

- elemental pure logical function equal_sp (a, b)
- elemental pure logical function equal_dp (a, b)

17.24.1 Detailed Description

Comparison of real values.

Compares two reals if they are numerically equal or not, i.e. equal:

$$|\frac{a-b}{b}| < \varepsilon$$

Parameters

in	*real(sp/dp) :: a*	First number to compare
in	*real(sp/dp) :: b*	Second number to compare

Returns

real(sp/dp) :: equal — $a == b$ logically true or false

Authors

Matthias Cuntz, Juliane Mai

Date

Feb 2014

17.24.2 Member Function/Subroutine Documentation

17.24.2.1 equal_dp()

```
elemental pure logical function mo_utils::equal::equal_dp (
             real(dp), intent(in) a,
             real(dp), intent(in) b )
```

17.24.2.2 equal_sp()

```
elemental pure logical function mo_utils::equal::equal_sp (
             real(sp), intent(in) a,
             real(sp), intent(in) b )
```

The documentation for this interface was generated from the following file:

- mo_utils.f90

17.25 mo_optimization_utils::eval_interface Interface Reference

Public Member Functions

- subroutine eval_interface (parameterset, runoff, sm_opti, basin_avg_tws, neutrons_opti, et_opti)

17.25.1 Constructor & Destructor Documentation

17.25.1.1 eval_interface()

```
subroutine mo_optimization_utils::eval_interface::eval_interface (
            real(dp), dimension(:), intent(in) parameterset,
            real(dp), dimension(:, :), intent(out), optional, allocatable runoff,
            real(dp), dimension(:, :), intent(out), optional, allocatable sm_opti,
            real(dp), dimension(:, :), intent(out), optional, allocatable basin_avg_tws,
            real(dp), dimension(:, :), intent(out), optional, allocatable neutrons_opti,
            real(dp), dimension(:, :), intent(out), optional, allocatable et_opti )
```

References mo_kind::dp.

The documentation for this interface was generated from the following file:

- mo_optimization_utils.f90

17.26 mo_orderpack::fndnth Interface Reference

Public Member Functions

- real(kind=dp) function d_fndnth (XDONT, NORD)
- real(kind=sp) function r_fndnth (XDONT, NORD)
- integer(kind=i4) function i_fndnth (XDONT, NORD)

17.26.1 Member Function/Subroutine Documentation

17.26.1.1 d_fndnth()

```
real(kind = dp) function mo_orderpack::fndnth::d_fndnth (
            real(kind = dp), dimension (:), intent(in) XDONT,
            integer(kind = i4), intent(in) NORD )
```

17.26.1.2 i_fndnth()

```
integer(kind = i4) function mo_orderpack::fndnth::i_fndnth (
            integer(kind = i4), dimension (:), intent(in) XDONT,
            integer(kind = i4), intent(in) NORD )
```

17.26.1.3 r_fndnth()

```
real(kind = sp) function mo_orderpack::fndnth::r_fndnth (
            real(kind = sp), dimension (:), intent(in) XDONT,
            integer(kind = i4), intent(in) NORD )
```

The documentation for this interface was generated from the following file:

- mo_orderpack.f90

17.27 mo_corr::four1 Interface Reference

Private Member Functions

- subroutine four1_sp (data, isign)
- subroutine four1_dp (data, isign)

17.27.1 Member Function/Subroutine Documentation

17.27.1.1 four1_dp()

```
subroutine mo_corr::four1::four1_dp (
            complex(dpc), dimension(:), intent(inout) data,
            integer(i4), intent(in) isign )  [private]
```

17.27.1.2 four1_sp()

```
subroutine mo_corr::four1::four1_sp (
            complex(spc), dimension(:), intent(inout) data,
            integer(i4), intent(in) isign )  [private]
```

The documentation for this interface was generated from the following file:

- mo_corr.f90

17.28 mo_corr::fourrow Interface Reference

Private Member Functions

- subroutine fourrow_sp (data, isign)
- subroutine fourrow_dp (data, isign)

17.28.1 Member Function/Subroutine Documentation

17.28.1.1 fourrow_dp()

```
subroutine mo_corr::fourrow::fourrow_dp (
            complex(dpc), dimension(:, :), intent(inout) data,
            integer(i4), intent(in) isign )  [private]
```

17.28.1.2 fourrow_sp()

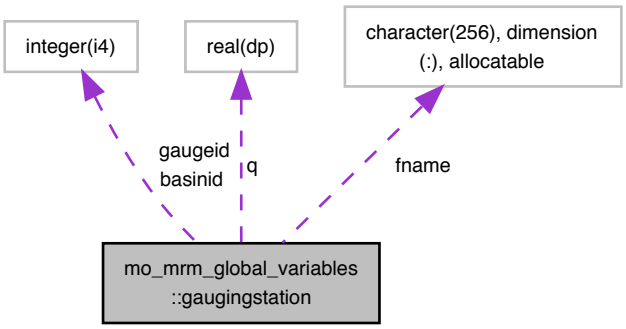
```
subroutine mo_corr::fourrow::fourrow_sp (
            complex(spc), dimension(:, :), intent(inout) data,
            integer(i4), intent(in) isign )  [private]
```

The documentation for this interface was generated from the following file:

- mo_corr.f90

17.29 mo_mrm_global_variables::gaugingstation Type Reference

Collaboration diagram for mo_mrm_global_variables::gaugingstation:



Public Attributes

- integer(i4), dimension(:), allocatable basinid
- integer(i4), dimension(:), allocatable gaugeid
- character(256), dimension(:), allocatable fname
- real(dp), dimension(:, :), allocatable q

17.29.1 Member Data Documentation

17.29.1.1 basinid

```
integer(i4), dimension(:), allocatable mo_mrm_global_variables::gaugingstation::basinid
```

17.29.1.2 fname

```
character(256), dimension(:), allocatable mo_mrm_global_variables::gaugingstation::fname
```

17.29.1.3 gaugeid

```
integer(i4), dimension(:), allocatable mo_mrm_global_variables::gaugingstation::gaugeid
```

17.29.1.4 q

```
real(dp), dimension(:, :), allocatable mo_mrm_global_variables::gaugingstation::q
```

The documentation for this type was generated from the following file:

- mo_mrm_global_variables.f90

17.30 mo_utils::ge Interface Reference

Public Member Functions

- elemental pure logical function greaterequal_sp (a, b)
- elemental pure logical function greaterequal_dp (a, b)

17.30.1 Member Function/Subroutine Documentation

17.30.1.1 greaterequal_dp()

```
elemental pure logical function mo_utils::ge::greaterequal_dp (
            real(dp), intent(in) a,
            real(dp), intent(in) b )
```

17.30.1.2 greaterequal_sp()

```
elemental pure logical function mo_utils::ge::greaterequal_sp (
            real(sp), intent(in) a,
            real(sp), intent(in) b )
```

The documentation for this interface was generated from the following file:

- mo_utils.f90

17.31 mo_anneal::generate_neighborhood_weight Interface Reference

Private Member Functions

- subroutine generate_neighborhood_weight_dp (truepara, cum_weight, save_state_xor, iTotalCounter, nIT↩ ERmax, neighborhood)

17.31.1 Member Function/Subroutine Documentation

17.31.1.1 generate_neighborhood_weight_dp()

```
subroutine mo_anneal::generate_neighborhood_weight::generate_neighborhood_weight_dp (
            integer(i4), dimension(:), intent(in) truepara,
            real(dp), dimension(:), intent(in) cum_weight,
            integer(i8), dimension(n_save_state), intent(inout) save_state_xor,
            integer(i4), intent(in) iTotalCounter,
            integer(i4), intent(in) nITERmax,
            logical, dimension(size(cum_weight)), intent(out) neighborhood )  [private]
```

The documentation for this interface was generated from the following file:

- mo_anneal.f90

17.32 mo_ncread::get_ncvar Interface Reference

Public Member Functions

- subroutine get_ncvar_0d_sp (Filename, VarName, Dat, fid)
- subroutine get_ncvar_0d_dp (Filename, VarName, Dat, fid)
- subroutine get_ncvar_1d_sp (Filename, VarName, Dat, start, a_count, fid)
- subroutine get_ncvar_1d_dp (Filename, VarName, Dat, start, a_count, fid)
- subroutine get_ncvar_2d_sp (Filename, VarName, Dat, start, a_count, fid)
- subroutine get_ncvar_2d_dp (Filename, VarName, Dat, start, a_count, fid)
- subroutine get_ncvar_3d_sp (Filename, VarName, Dat, start, a_count, fid)
- subroutine get_ncvar_3d_dp (Filename, VarName, Dat, start, a_count, fid)
- subroutine get_ncvar_4d_sp (Filename, VarName, Dat, start, a_count, fid)
- subroutine get_ncvar_4d_dp (Filename, VarName, Dat, start, a_count, fid)
- subroutine get_ncvar_5d_sp (Filename, VarName, Dat, start, a_count, fid)
- subroutine get_ncvar_5d_dp (Filename, VarName, Dat, start, a_count, fid)
- subroutine get_ncvar_0d_i4 (Filename, VarName, Dat, fid)
- subroutine get_ncvar_1d_i4 (Filename, VarName, Dat, start, a_count, fid)
- subroutine get_ncvar_2d_i4 (Filename, VarName, Dat, start, a_count, fid)
- subroutine get_ncvar_3d_i4 (Filename, VarName, Dat, start, a_count, fid)
- subroutine get_ncvar_4d_i4 (Filename, VarName, Dat, start, a_count, fid)
- subroutine get_ncvar_5d_i4 (Filename, VarName, Dat, start, a_count, fid)
- subroutine get_ncvar_0d_i1 (Filename, VarName, Dat, fid)
- subroutine get_ncvar_1d_i1 (Filename, VarName, Dat, start, a_count, fid)
- subroutine get_ncvar_2d_i1 (Filename, VarName, Dat, start, a_count, fid)
- subroutine get_ncvar_3d_i1 (Filename, VarName, Dat, start, a_count, fid)
- subroutine get_ncvar_4d_i1 (Filename, VarName, Dat, start, a_count, fid)
- subroutine get_ncvar_5d_i1 (Filename, VarName, Dat, start, a_count, fid)

17.32.1 Member Function/Subroutine Documentation

17.32.1.1 get_ncvar_0d_dp()

```
subroutine mo_ncread::get_ncvar::get_ncvar_0d_dp (
            character(len = *), intent(in) Filename,
            character(len = *), intent(in) VarName,
            real(dp), intent(inout) Dat,
            integer(i4), intent(in), optional fid )
```

17.32.1.2 get_ncvar_0d_i1()

```
subroutine mo_ncread::get_ncvar::get_ncvar_0d_i1 (
             character(len = *), intent(in) Filename,
             character(len = *), intent(in) VarName,
             integer(1), intent(inout) Dat,
             integer(i4), intent(in), optional fid )
```

17.32.1.3 get_ncvar_0d_i4()

```
subroutine mo_ncread::get_ncvar::get_ncvar_0d_i4 (
             character(len = *), intent(in) Filename,
             character(len = *), intent(in) VarName,
             integer(i4), intent(inout) Dat,
             integer(i4), intent(in), optional fid )
```

17.32.1.4 get_ncvar_0d_sp()

```
subroutine mo_ncread::get_ncvar::get_ncvar_0d_sp (
             character(len = *), intent(in) Filename,
             character(len = *), intent(in) VarName,
             real(sp), intent(inout) Dat,
             integer(i4), intent(in), optional fid )
```

17.32.1.5 get_ncvar_1d_dp()

```
subroutine mo_ncread::get_ncvar::get_ncvar_1d_dp (
             character(len = *), intent(in) Filename,
             character(len = *), intent(in) VarName,
             real(dp), dimension(:), intent(inout), allocatable Dat,
             integer(i4), dimension(:), intent(in), optional start,
             integer(i4), dimension(:), intent(in), optional a_count,
             integer(i4), intent(in), optional fid )
```

17.32.1.6 get_ncvar_1d_i1()

```
subroutine mo_ncread::get_ncvar::get_ncvar_1d_i1 (
             character(len = *), intent(in) Filename,
             character(len = *), intent(in) VarName,
             integer(1), dimension(:), intent(inout), allocatable Dat,
             integer(i4), dimension(:), intent(in), optional start,
             integer(i4), dimension(:), intent(in), optional a_count,
             integer(i4), intent(in), optional fid )
```

17.32.1.7 get_ncvar_1d_i4()

```
subroutine mo_ncread::get_ncvar::get_ncvar_1d_i4 (
```

```
            character(len = *), intent(in) Filename,
            character(len = *), intent(in) VarName,
            integer(i4), dimension(:), intent(inout), allocatable Dat,
            integer(i4), dimension(:), intent(in), optional start,
            integer(i4), dimension(:), intent(in), optional a_count,
            integer(i4), intent(in), optional fid )
```

17.32.1.8 get_ncvar_1d_sp()

```
subroutine mo_ncread::get_ncvar::get_ncvar_1d_sp (
            character(len = *), intent(in) Filename,
            character(len = *), intent(in) VarName,
            real(sp), dimension(:), intent(inout), allocatable Dat,
            integer(i4), dimension(:), intent(in), optional start,
            integer(i4), dimension(:), intent(in), optional a_count,
            integer(i4), intent(in), optional fid )
```

17.32.1.9 get_ncvar_2d_dp()

```
subroutine mo_ncread::get_ncvar::get_ncvar_2d_dp (
            character(len = *), intent(in) Filename,
            character(len = *), intent(in) VarName,
            real(dp), dimension(:, :), intent(inout), allocatable Dat,
            integer(i4), dimension(:), intent(in), optional start,
            integer(i4), dimension(:), intent(in), optional a_count,
            integer(i4), intent(in), optional fid )
```

17.32.1.10 get_ncvar_2d_i1()

```
subroutine mo_ncread::get_ncvar::get_ncvar_2d_i1 (
            character(len = *), intent(in) Filename,
            character(len = *), intent(in) VarName,
            integer(1), dimension(:, :), intent(inout), allocatable Dat,
            integer(i4), dimension(:), intent(in), optional start,
            integer(i4), dimension(:), intent(in), optional a_count,
            integer(i4), intent(in), optional fid )
```

17.32.1.11 get_ncvar_2d_i4()

```
subroutine mo_ncread::get_ncvar::get_ncvar_2d_i4 (
            character(len = *), intent(in) Filename,
            character(len = *), intent(in) VarName,
            integer(i4), dimension(:, :), intent(inout), allocatable Dat,
            integer(i4), dimension(:), intent(in), optional start,
            integer(i4), dimension(:), intent(in), optional a_count,
            integer(i4), intent(in), optional fid )
```

17.32.1.12 get_ncvar_2d_sp()

```
subroutine mo_ncread::get_ncvar::get_ncvar_2d_sp (
             character(len = *), intent(in) Filename,
             character(len = *), intent(in) VarName,
             real(sp), dimension(:, :), intent(inout), allocatable Dat,
             integer(i4), dimension(:), intent(in), optional start,
             integer(i4), dimension(:), intent(in), optional a_count,
             integer(i4), intent(in), optional fid )
```

17.32.1.13 get_ncvar_3d_dp()

```
subroutine mo_ncread::get_ncvar::get_ncvar_3d_dp (
             character(len = *), intent(in) Filename,
             character(len = *), intent(in) VarName,
             real(dp), dimension(:, :, :), intent(inout), allocatable Dat,
             integer(i4), dimension(:), intent(in), optional start,
             integer(i4), dimension(:), intent(in), optional a_count,
             integer(i4), intent(in), optional fid )
```

17.32.1.14 get_ncvar_3d_i1()

```
subroutine mo_ncread::get_ncvar::get_ncvar_3d_i1 (
             character(len = *), intent(in) Filename,
             character(len = *), intent(in) VarName,
             integer(1), dimension(:, :, :), intent(inout), allocatable Dat,
             integer(i4), dimension(:), intent(in), optional start,
             integer(i4), dimension(:), intent(in), optional a_count,
             integer(i4), intent(in), optional fid )
```

17.32.1.15 get_ncvar_3d_i4()

```
subroutine mo_ncread::get_ncvar::get_ncvar_3d_i4 (
             character(len = *), intent(in) Filename,
             character(len = *), intent(in) VarName,
             integer(i4), dimension(:, :, :), intent(inout), allocatable Dat,
             integer(i4), dimension(:), intent(in), optional start,
             integer(i4), dimension(:), intent(in), optional a_count,
             integer(i4), intent(in), optional fid )
```

17.32.1.16 get_ncvar_3d_sp()

```
subroutine mo_ncread::get_ncvar::get_ncvar_3d_sp (
             character(len = *), intent(in) Filename,
             character(len = *), intent(in) VarName,
             real(sp), dimension(:, :, :), intent(inout), allocatable Dat,
             integer(i4), dimension(:), intent(in), optional start,
             integer(i4), dimension(:), intent(in), optional a_count,
             integer(i4), intent(in), optional fid )
```

17.32.1.17 get_ncvar_4d_dp()

```
subroutine mo_ncread::get_ncvar::get_ncvar_4d_dp (
             character(len = *), intent(in) Filename,
             character(len = *), intent(in) VarName,
             real(dp), dimension(:, :, :, :), intent(inout), allocatable Dat,
             integer(i4), dimension(:), intent(in), optional start,
             integer(i4), dimension(:), intent(in), optional a_count,
             integer(i4), intent(in), optional fid )
```

17.32.1.18 get_ncvar_4d_i1()

```
subroutine mo_ncread::get_ncvar::get_ncvar_4d_i1 (
             character(len = *), intent(in) Filename,
             character(len = *), intent(in) VarName,
             integer(1), dimension(:, :, :, :), intent(inout), allocatable Dat,
             integer(i4), dimension(:), intent(in), optional start,
             integer(i4), dimension(:), intent(in), optional a_count,
             integer(i4), intent(in), optional fid )
```

17.32.1.19 get_ncvar_4d_i4()

```
subroutine mo_ncread::get_ncvar::get_ncvar_4d_i4 (
             character(len = *), intent(in) Filename,
             character(len = *), intent(in) VarName,
             integer(i4), dimension(:, :, :, :), intent(inout), allocatable Dat,
             integer(i4), dimension(:), intent(in), optional start,
             integer(i4), dimension(:), intent(in), optional a_count,
             integer(i4), intent(in), optional fid )
```

17.32.1.20 get_ncvar_4d_sp()

```
subroutine mo_ncread::get_ncvar::get_ncvar_4d_sp (
             character(len = *), intent(in) Filename,
             character(len = *), intent(in) VarName,
             real(sp), dimension(:, :, :, :), intent(inout), allocatable Dat,
             integer(i4), dimension(:), intent(in), optional start,
             integer(i4), dimension(:), intent(in), optional a_count,
             integer(i4), intent(in), optional fid )
```

17.32.1.21 get_ncvar_5d_dp()

```
subroutine mo_ncread::get_ncvar::get_ncvar_5d_dp (
             character(len = *), intent(in) Filename,
             character(len = *), intent(in) VarName,
             real(dp), dimension(:, :, :, :, :), intent(inout), allocatable Dat,
```

```
            integer(i4), dimension(:), intent(in), optional start,
            integer(i4), dimension(:), intent(in), optional a_count,
            integer(i4), intent(in), optional fid )
```

17.32.1.22 get_ncvar_5d_i1()

```
subroutine mo_ncread::get_ncvar::get_ncvar_5d_i1 (
            character(len = *), intent(in) Filename,
            character(len = *), intent(in) VarName,
            integer(1), dimension(:, :, :, :, :), intent(inout), allocatable Dat,
            integer(i4), dimension(:), intent(in), optional start,
            integer(i4), dimension(:), intent(in), optional a_count,
            integer(i4), intent(in), optional fid )
```

17.32.1.23 get_ncvar_5d_i4()

```
subroutine mo_ncread::get_ncvar::get_ncvar_5d_i4 (
            character(len = *), intent(in) Filename,
            character(len = *), intent(in) VarName,
            integer(i4), dimension(:, :, :, :, :), intent(inout), allocatable Dat,
            integer(i4), dimension(:), intent(in), optional start,
            integer(i4), dimension(:), intent(in), optional a_count,
            integer(i4), intent(in), optional fid )
```

17.32.1.24 get_ncvar_5d_sp()

```
subroutine mo_ncread::get_ncvar::get_ncvar_5d_sp (
            character(len = *), intent(in) Filename,
            character(len = *), intent(in) VarName,
            real(sp), dimension(:, :, :, :, :), intent(inout), allocatable Dat,
            integer(i4), dimension(:), intent(in), optional start,
            integer(i4), dimension(:), intent(in), optional a_count,
            integer(i4), intent(in), optional fid )
```

The documentation for this interface was generated from the following file:

- mo_ncread.f90

17.33 mo_xor4096::get_timeseed Interface Reference

Public Member Functions

- subroutine get_timeseed_i4_0d (seed)
- subroutine get_timeseed_i4_1d (seed)
- subroutine get_timeseed_i8_0d (seed)
- subroutine get_timeseed_i8_1d (seed)

17.33.1 Member Function/Subroutine Documentation

17.33.1.1 get_timeseed_i4_0d()

```
subroutine mo_xor4096::get_timeseed::get_timeseed_i4_0d (
             integer(i4), intent(inout) seed )
```

17.33.1.2 get_timeseed_i4_1d()

```
subroutine mo_xor4096::get_timeseed::get_timeseed_i4_1d (
             integer(i4), dimension(:), intent(inout) seed )
```

17.33.1.3 get_timeseed_i8_0d()

```
subroutine mo_xor4096::get_timeseed::get_timeseed_i8_0d (
             integer(i8), intent(inout) seed )
```

17.33.1.4 get_timeseed_i8_1d()

```
subroutine mo_xor4096::get_timeseed::get_timeseed_i8_1d (
             integer(i8), dimension(:), intent(inout) seed )
```

The documentation for this interface was generated from the following file:

- mo_xor4096.f90

17.34 mo_anneal::gettemperature Interface Reference

GetTemperature.

Public Member Functions

- real(dp) function gettemperature_dp (paraset, cost, eval, acc_goal, prange, prange_func, samplesize, maskpara, seeds, printflag, weight, maxit, undef_funcval)

17.34.1 Detailed Description

GetTemperature.

Determines an initial temperature for Simulated Annealing achieving

Parameters

in	*REAL(DP), DIMENSION(:) :: paraset*	initial (valid) parameter set
in	*INTERFACE :: cost_dp*	interface calculating the cost function at a given point
in	*REAL(DP) :: acc_goal*	Acceptance Ratio which has to be achieved
in	*REAL(DP), DIMENSION(size(para),2), optional :: prange*	lower and upper bound per parameter

Parameters

in	*INTERFACE, optional :: prange_func*	interface calculating the feasible range for a parameter at a certain point, if ranges are variable
in	*INTEGER(I4), optional :: samplesize*	number of iterations the estimation of temperature is based on DEFAULT: Max(20_i4∗n,250_i4)
in	*LOGICAL, DIMENSION(size(para)), optional :: maskpara*	maskpara(i) = .true. −> parameter is optimized maskpara(i) = .false. −> parameter is discarded from optimiztaion DEFAULT: .true.
in	*INTEGER(I4/I8), DIMENSION(2), optional :: seeds*	Seeds of random numbers used for random parameter set generation DEFAULT: dependent on current time
in	*LOGICAL, optional :: printflag*	If .true. detailed command line output is written DEFAULT: .false.
in	*REAL(DP), DIMENSION(size(para,1)), optional :: weight*	vector of weights per parameter gives the frequency of parameter to be chosen for optimization (will be scaled to a CDF internally) eg. [1,2,1] −> parameter 2 is chosen twice as often as parameter 1 and 2
in	*LOGICAL, optional :: maxit*	minimizing (.false.) or maximizing (.true.) a function DEFAULT: .false. (minimization)
in	*REAL(DP), optional :: undef_funcval*	objective function value defining invalid model output, e.g. -999.0_dp

Returns

real(dp) :: temperature — Temperature achieving a certain acceptance ratio in Simulated Annealing

Note

Either fixed parameter range (prange) OR flexible parameter range (function interface prange_func) has to be given in calling sequence.
Only double precision version available. If single precision is needed not only DP has to be replaced by SP but also I8 of save_state (random number variables) has to be replaced by I4.
ParaChangeMode > 1 is not applied in GetTemperature. For Temperature estimation always only one single parameter is changed (ParaChangeMode=1) which should give theoretically always the best estimate.
Cost and prange_func are user defined functions. See interface definition.

17.34.2 Member Function/Subroutine Documentation

17.34.2.1 gettemperature_dp()

```
real(dp) function mo_anneal::gettemperature::gettemperature_dp (
            real(dp), dimension(:), intent(in) paraset,
            procedure(objective_interface), intent(in), pointer cost,
            procedure(eval_interface), intent(in), pointer eval,
            real(dp), intent(in) acc_goal,
            real(dp), dimension(size(paraset, 1), 2), intent(in), optional prange,
            optional prange_func,
```

```
            integer(i4), intent(in), optional samplesize,
            logical, dimension(size(paraset, 1)), intent(in), optional maskpara,
            integer(i8), dimension(2), intent(in), optional seeds,
            logical, intent(in), optional printflag,
            real(dp), dimension(size(paraset, 1)), intent(in), optional weight,
            logical, intent(in), optional maxit,
            real(dp), intent(in), optional undef_funcval )
```

The documentation for this interface was generated from the following file:

- mo_anneal.f90

17.35 mo_utils::greaterequal Interface Reference

Public Member Functions

- elemental pure logical function greaterequal_sp (a, b)
- elemental pure logical function greaterequal_dp (a, b)

17.35.1 Member Function/Subroutine Documentation

17.35.1.1 greaterequal_dp()

```
elemental pure logical function mo_utils::greaterequal::greaterequal_dp (
            real(dp), intent(in) a,
            real(dp), intent(in) b )
```

17.35.1.2 greaterequal_sp()

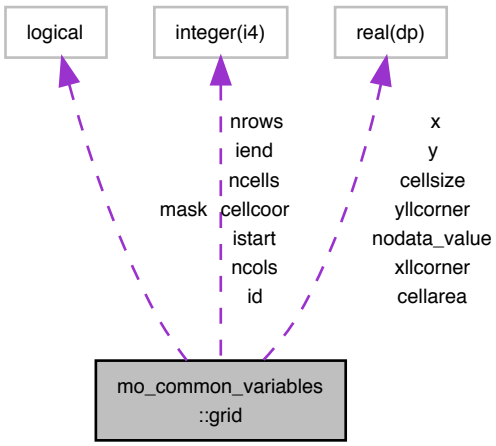
```
elemental pure logical function mo_utils::greaterequal::greaterequal_sp (
            real(sp), intent(in) a,
            real(sp), intent(in) b )
```

The documentation for this interface was generated from the following file:

- mo_utils.f90

17.36 mo_common_variables::grid Type Reference

Collaboration diagram for mo_common_variables::grid:



Public Attributes

- integer(i4) ncols
- integer(i4) nrows
- integer(i4) ncells
- real(dp) xllcorner
- real(dp) yllcorner
- real(dp) cellsize
- real(dp) nodata_value
- real(dp), dimension(:, :), allocatable x
- real(dp), dimension(:, :), allocatable y
- logical, dimension(:, :), allocatable mask
- integer(i4) istart
- integer(i4) iend
- integer(i4), dimension(:, :), allocatable cellcoor
- real(dp), dimension(:), allocatable cellarea
- integer(i4), dimension(:), allocatable id

17.36.1 Member Data Documentation

17.36.1.1 cellarea

```
real(dp), dimension(:), allocatable mo_common_variables::grid::cellarea
```

17.36.1.2 cellcoor

```
integer(i4), dimension(:, :), allocatable mo_common_variables::grid::cellcoor
```

17.36.1.3 cellsize

```
real(dp) mo_common_variables::grid::cellsize
```

17.36.1.4 id

```
integer(i4), dimension(:), allocatable mo_common_variables::grid::id
```

17.36.1.5 iend

```
integer(i4) mo_common_variables::grid::iend
```

17.36.1.6 istart

```
integer(i4) mo_common_variables::grid::istart
```

17.36.1.7 mask

```
logical, dimension(:, :), allocatable mo_common_variables::grid::mask
```

17.36.1.8 ncells

```
integer(i4) mo_common_variables::grid::ncells
```

17.36.1.9 ncols

```
integer(i4) mo_common_variables::grid::ncols
```

17.36.1.10 nodata_value

```
real(dp) mo_common_variables::grid::nodata_value
```

17.36.1.11 nrows

```
integer(i4) mo_common_variables::grid::nrows
```

17.36.1.12 x

```
real(dp), dimension(:, :), allocatable mo_common_variables::grid::x
```

17.36.1.13 xllcorner

```
real(dp) mo_common_variables::grid::xllcorner
```

17.36.1.14 y

```
real(dp), dimension(:, :), allocatable mo_common_variables::grid::y
```

17.36.1.15 yllcorner

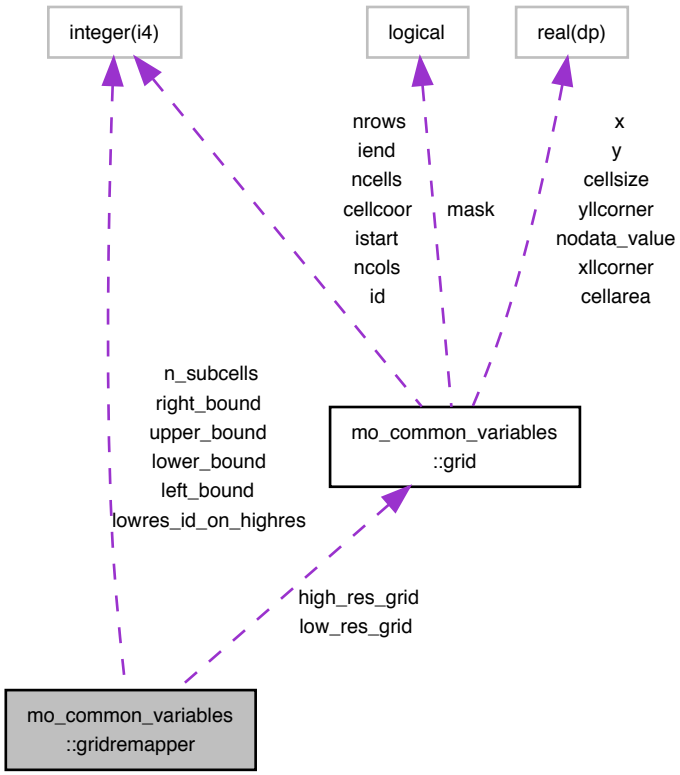
```
real(dp) mo_common_variables::grid::yllcorner
```

The documentation for this type was generated from the following file:

- mo_common_variables.f90

17.37 mo_common_variables::gridremapper Type Reference

Collaboration diagram for mo_common_variables::gridremapper:



Public Attributes

- type(grid), pointer high_res_grid
- type(grid), pointer low_res_grid
- integer(i4), dimension(:), allocatable lower_bound
- integer(i4), dimension(:), allocatable upper_bound
- integer(i4), dimension(:), allocatable left_bound
- integer(i4), dimension(:), allocatable right_bound
- integer(i4), dimension(:), allocatable n_subcells
- integer(i4), dimension(:, :), allocatable lowres_id_on_highres

17.37.1 Member Data Documentation

17.37.1.1 high_res_grid

```
type(grid), pointer mo_common_variables::gridremapper::high_res_grid
```

17.37.1.2 left_bound

```
integer(i4), dimension(:), allocatable mo_common_variables::gridremapper::left_bound
```

17.37.1.3 low_res_grid

```
type(grid), pointer mo_common_variables::gridremapper::low_res_grid
```

17.37.1.4 lower_bound

```
integer(i4), dimension(:), allocatable mo_common_variables::gridremapper::lower_bound
```

17.37.1.5 lowres_id_on_highres

```
integer(i4), dimension(:, :), allocatable mo_common_variables::gridremapper::lowres_id_on_↩
highres
```

17.37.1.6 n_subcells

```
integer(i4), dimension(:), allocatable mo_common_variables::gridremapper::n_subcells
```

17.37.1.7 right_bound

```
integer(i4), dimension(:), allocatable mo_common_variables::gridremapper::right_bound
```

17.37.1.8 upper_bound

```
integer(i4), dimension(:), allocatable mo_common_variables::gridremapper::upper_bound
```

The documentation for this type was generated from the following file:

- mo_common_variables.f90

17.38 mo_temporal_aggregation::hour2day_average Interface Reference

Hour-to-day average (hour2day_average)

Public Member Functions

- subroutine hour2day_average_dp (hourly_data, yearS, monthS, dayS, hourS, day_avg, misval, rm_misval)

17.38.1 Detailed Description

Hour-to-day average (hour2day_average)

converts hourly time series to daily

Parameters

in	*real(sp/dp) :: hourly_data(:)*	array of hourly time series
in	*integer(i4) :: year*	year of the starting time
in	*integer(i4) :: month*	month of the starting time
in	*integer(i4) :: day*	day of the starting time
in	*integer(i4) :: hour*	hour of the starting time
in	*real(sp/dp) :: day_average(:)*	array of daily averaged values
in	*real(sp/dp) :: misval*	missing value definition
in	*logical :: rm_misval*	switch to exclude missing values

Note

Hours values should be from 0 to 23 (NOT from 1 to 24!)

Author

Oldrich Rakovec, Rohini Kumar

Date

Oct 2015

17.38.2 Member Function/Subroutine Documentation

17.38.2.1 hour2day_average_dp()

```
subroutine mo_temporal_aggregation::hour2day_average::hour2day_average_dp (
             real(dp), dimension(:), intent(in) hourly_data,
             integer(i4), intent(in) yearS,
             integer(i4), intent(in) monthS,
             integer(i4), intent(in) dayS,
             integer(i4), intent(in) hourS,
             real(dp), dimension(:), intent(inout), allocatable day_avg,
             real(dp), intent(in), optional misval,
             logical, intent(in), optional rm_misval )
```

The documentation for this interface was generated from the following file:

- mo_temporal_aggregation.f90

17.39 mo_orderpack::indmed Interface Reference

Public Member Functions

- subroutine d_indmed (XDONT, INDM)

- subroutine r_indmed (XDONT, INDM)
- subroutine i_indmed (XDONT, INDM)

17.39.1 Member Function/Subroutine Documentation

17.39.1.1 d_indmed()

```
subroutine mo_orderpack::indmed::d_indmed (
             real(kind = dp), dimension (:), intent(in) XDONT,
             integer(kind = i4), intent(out) INDM )
```

17.39.1.2 i_indmed()

```
subroutine mo_orderpack::indmed::i_indmed (
             integer(kind = i4), dimension (:), intent(in) XDONT,
             integer(kind = i4), intent(out) INDM )
```

17.39.1.3 r_indmed()

```
subroutine mo_orderpack::indmed::r_indmed (
             real(kind = sp), dimension (:), intent(in) XDONT,
             integer(kind = i4), intent(out) INDM )
```

The documentation for this interface was generated from the following file:

- mo_orderpack.f90

17.40 mo_orderpack::indnth Interface Reference

Public Member Functions

- integer(kind=i4) function d_indnth (XDONT, NORD)
- integer(kind=i4) function r_indnth (XDONT, NORD)
- integer(kind=i4) function i_indnth (XDONT, NORD)

17.40.1 Member Function/Subroutine Documentation

17.40.1.1 d_indnth()

```
integer(kind = i4) function mo_orderpack::indnth::d_indnth (
             real(kind = dp), dimension (:), intent(in) XDONT,
             integer(kind = i4), intent(in) NORD )
```

17.40.1.2 i_indnth()

```
integer(kind = i4) function mo_orderpack::indnth::i_indnth (
             integer(kind = i4), dimension (:), intent(in) XDONT,
             integer(kind = i4), intent(in) NORD )
```

17.40.1.3 r_indnth()

```
integer(kind = i4) function mo_orderpack::indnth::r_indnth (
             real(kind = sp), dimension (:), intent(in) XDONT,
             integer(kind = i4), intent(in) NORD )
```

The documentation for this interface was generated from the following file:

- mo_orderpack.f90

17.41 mo_orderpack::inspar Interface Reference

Public Member Functions

- subroutine d_inspar (XDONT, NORD)
- subroutine r_inspar (XDONT, NORD)
- subroutine i_inspar (XDONT, NORD)

17.41.1 Member Function/Subroutine Documentation

17.41.1.1 d_inspar()

```
subroutine mo_orderpack::inspar::d_inspar (
             real(kind = dp), dimension (:), intent(inout) XDONT,
             integer(kind = i4), intent(in) NORD )
```

17.41.1.2 i_inspar()

```
subroutine mo_orderpack::inspar::i_inspar (
             integer(kind = i4), dimension (:), intent(inout) XDONT,
             integer(kind = i4), intent(in) NORD )
```

17.41.1.3 r_inspar()

```
subroutine mo_orderpack::inspar::r_inspar (
             real(kind = sp), dimension (:), intent(inout) XDONT,
             integer(kind = i4), intent(in) NORD )
```

The documentation for this interface was generated from the following file:

- mo_orderpack.f90

17.42 mo_orderpack::inssor Interface Reference

Public Member Functions

- subroutine d_inssor (XDONT)
- subroutine r_inssor (XDONT)
- subroutine i_inssor (XDONT)

17.42.1 Member Function/Subroutine Documentation

17.42.1.1 d_inssor()

```
subroutine mo_orderpack::inssor::d_inssor (
             real(kind = dp), dimension (:), intent(inout) XDONT )
```

17.42.1.2 i_inssor()

```
subroutine mo_orderpack::inssor::i_inssor (
             integer(kind = i4), dimension (:), intent(inout) XDONT )
```

17.42.1.3 r_inssor()

```
subroutine mo_orderpack::inssor::r_inssor (
             real(kind = sp), dimension (:), intent(inout) XDONT )
```

The documentation for this interface was generated from the following file:

- mo_orderpack.f90

17.43 mo_utils::is_finite Interface Reference

.true. if not IEEE Inf, IEEE NaN, nor IEEE Inf nor IEEE NaN, respectively.

Public Member Functions

- elemental pure logical function is_finite_sp (a)
- elemental pure logical function is_finite_dp (a)

17.43.1 Detailed Description

.true. if not IEEE Inf, IEEE NaN, nor IEEE Inf nor IEEE NaN, respectively.

Checks for IEEE Inf and IEEE NaN, i.e. Infinity and Not-a-Number.
Wraps to functions of the intrinsic module ieee_arithmetic but gives alternatives for gfortran, which does not provide ieee_arithmetic.

Parameters

in	*real(sp/dp) :: x*	Number to check

Returns

logical :: is_finite/is_nan/is_normal — $a/=Inf, a==NaN, a/=Inf and a==NaN$, logically true or false

Authors

Matthias Cuntz

Date

Mar 2015

17.43.2 Member Function/Subroutine Documentation

17.43.2.1 is_finite_dp()

```
elemental pure logical function mo_utils::is_finite::is_finite_dp (
             real(dp), intent(in) a )
```

17.43.2.2 is_finite_sp()

```
elemental pure logical function mo_utils::is_finite::is_finite_sp (
             real(sp), intent(in) a )
```

The documentation for this interface was generated from the following file:

- mo_utils.f90

17.44 mo_utils::is_nan Interface Reference

Public Member Functions

- elemental pure logical function is_nan_sp (a)
- elemental pure logical function is_nan_dp (a)

17.44.1 Member Function/Subroutine Documentation

17.44.1.1 is_nan_dp()

```
elemental pure logical function mo_utils::is_nan::is_nan_dp (
             real(dp), intent(in) a )
```

17.44.1.2 is_nan_sp()

```
elemental pure logical function mo_utils::is_nan::is_nan_sp (
            real(sp), intent(in) a )
```

The documentation for this interface was generated from the following file:

- mo_utils.f90

17.45 mo_utils::is_normal Interface Reference

Public Member Functions

- elemental pure logical function is_normal_sp (a)
- elemental pure logical function is_normal_dp (a)

17.45.1 Member Function/Subroutine Documentation

17.45.1.1 is_normal_dp()

```
elemental pure logical function mo_utils::is_normal::is_normal_dp (
            real(dp), intent(in) a )
```

17.45.1.2 is_normal_sp()

```
elemental pure logical function mo_utils::is_normal::is_normal_sp (
            real(sp), intent(in) a )
```

The documentation for this interface was generated from the following file:

- mo_utils.f90

17.46 mo_errormeasures::kge Interface Reference

Kling-Gupta-Efficiency measure.

Public Member Functions

- real(dp) function kge_dp_1d (x, y, mask)
- real(dp) function kge_dp_2d (x, y, mask)
- real(dp) function kge_dp_3d (x, y, mask)
- real(sp) function kge_sp_1d (x, y, mask)
- real(sp) function kge_sp_2d (x, y, mask)
- real(sp) function kge_sp_3d (x, y, mask)

17.46.1 Detailed Description

Kling-Gupta-Efficiency measure.

The Kling-Gupta model efficiency coefficient KGE is

$$KGE = 1 - \sqrt{((1-r)^2 + (1-\alpha)^2 + (1-\beta)^2)}$$

where
r = Pearson product-moment correlation coefficient
α = ratio of simulated mean to observed mean
β = ratio of simulated standard deviation to observed standard deviation
This three measures are calculated between two arrays (1d, 2d, or 3d). Usually, one is an observation and the second is a modelled variable.
The higher the KGE the better the observation and simulation are matching. The upper limit of KGE is 1.
Therefore, if you apply a minimization algorithm to calibrate regarding KGE you have to use the objective function

$$obj\_value = 1.0 - KGE$$

which has then the optimum at 0.0. (Like for the NSE where you always optimize 1-NSE.)
real(sp/dp), dimension(:) :: x, y 1D-array with input numbers real(sp/dp), dimension(:,:) :: x, y 2D-array with input numbers real(sp/dp), dimension(:,:,:) :: x, y 3D-array with input numbers logical :: mask(:) 1D-array of logical values with size(x/y). logical :: mask(:,:) 2D-array of logical values with size(x/y). logical :: mask(:,:,:) 3D-array of logical values with size(x/y).

Returns

kge — Kling-Gupta-Efficiency (value less equal 1.0)

Note

Input values must be floating points.
Gupta, Hoshin V., et al. "Decomposition of the mean squared error and NSE performance criteria: Implications for improving hydrological modelling." Journal of Hydrology 377.1 (2009): 80-91.

Author

Rohini Kumar

Date

August 2014

17.46.2 Member Function/Subroutine Documentation

17.46.2.1 kge_dp_1d()

```
real(dp) function mo_errormeasures::kge::kge_dp_1d (
             real(dp), dimension(:), intent(in) x,
             real(dp), dimension(:), intent(in) y,
             logical, dimension(:), intent(in), optional mask )
```

17.46.2.2 kge_dp_2d()

```
real(dp) function mo_errormeasures::kge::kge_dp_2d (
             real(dp), dimension(:, :), intent(in) x,
             real(dp), dimension(:, :), intent(in) y,
             logical, dimension(:, :), intent(in), optional mask )
```

17.46.2.3 kge_dp_3d()

```
real(dp) function mo_errormeasures::kge::kge_dp_3d (
             real(dp), dimension(:, :, :), intent(in) x,
             real(dp), dimension(:, :, :), intent(in) y,
             logical, dimension(:, :, :), intent(in), optional mask )
```

17.46.2.4 kge_sp_1d()

```
real(sp) function mo_errormeasures::kge::kge_sp_1d (
             real(sp), dimension(:), intent(in) x,
             real(sp), dimension(:), intent(in) y,
             logical, dimension(:), intent(in), optional mask )
```

17.46.2.5 kge_sp_2d()

```
real(sp) function mo_errormeasures::kge::kge_sp_2d (
             real(sp), dimension(:, :), intent(in) x,
             real(sp), dimension(:, :), intent(in) y,
             logical, dimension(:, :), intent(in), optional mask )
```

17.46.2.6 kge_sp_3d()

```
real(sp) function mo_errormeasures::kge::kge_sp_3d (
             real(sp), dimension(:, :, :), intent(in) x,
             real(sp), dimension(:, :, :), intent(in) y,
             logical, dimension(:, :, :), intent(in), optional mask )
```

The documentation for this interface was generated from the following file:

- mo_errormeasures.f90

17.47 mo_errormeasures::kgenocorr Interface Reference

Kling-Gupta-Efficiency measure without correlation.

Public Member Functions

- real(dp) function kgenocorr_dp_1d (x, y, mask)
- real(dp) function kgenocorr_dp_2d (x, y, mask)

- real(dp) function kgenocorr_dp_3d (x, y, mask)
- real(sp) function kgenocorr_sp_1d (x, y, mask)
- real(sp) function kgenocorr_sp_2d (x, y, mask)
- real(sp) function kgenocorr_sp_3d (x, y, mask)

17.47.1 Detailed Description

Kling-Gupta-Efficiency measure without correlation.

The modified Kling-Gupta model efficiency coefficient $KGEnocorr$ is

$$KGEnocorr = 1 - \sqrt{((1-\alpha)^2 + (1-\beta)^2)}$$

where
α = ratio of simulated mean to observed mean
β = ratio of simulated standard deviation to observed standard deviation
This two measures are calculated between two arrays (1d, 2d, or 3d). Usually, one is an observation and the second is a modelled variable.
The higher the KGEnocorr the better the observation and simulation are matching. The upper limit of KGEnocorr is 1.
Therefore, if you apply a minimization algorithm to calibrate regarding KGEnocorr you have to use the objective function

$$obj\_value = 1.0 - KGEnocorr$$

which has then the optimum at 0.0. (Like for the NSE where you always optimize 1-NSE.)
real(sp/dp), dimension(:) :: x, y 1D-array with input numbers real(sp/dp), dimension(:,:) :: x, y 2D-array with input numbers real(sp/dp), dimension(:,:,:) :: x, y 3D-array with input numbers logical :: mask(:) 1D-array of logical values with size(x/y). logical :: mask(:,:) 2D-array of logical values with size(x/y). logical :: mask(:,:,:) 3D-array of logical values with size(x/y).

Returns

kgenocorr — Kling-Gupta-Efficiency without correlation (value less equal 1.0)

Note

Input values must be floating points.
Gupta, Hoshin V., et al. "Decomposition of the mean squared error and NSE performance criteria: Implications for improving hydrological modelling." Journal of Hydrology 377.1 (2009): 80-91.

Author

Rohini Kumar

Date

August 2014

17.47.2 Member Function/Subroutine Documentation

17.47.2.1 kgenocorr_dp_1d()

```
real(dp) function mo_errormeasures::kgenocorr::kgenocorr_dp_1d (
             real(dp), dimension(:), intent(in) x,
             real(dp), dimension(:), intent(in) y,
             logical, dimension(:), intent(in), optional mask )
```

17.47.2.2 kgenocorr_dp_2d()

```
real(dp) function mo_errormeasures::kgenocorr::kgenocorr_dp_2d (
             real(dp), dimension(:, :), intent(in) x,
             real(dp), dimension(:, :), intent(in) y,
             logical, dimension(:, :), intent(in), optional mask )
```

17.47.2.3 kgenocorr_dp_3d()

```
real(dp) function mo_errormeasures::kgenocorr::kgenocorr_dp_3d (
             real(dp), dimension(:, :, :), intent(in) x,
             real(dp), dimension(:, :, :), intent(in) y,
             logical, dimension(:, :, :), intent(in), optional mask )
```

17.47.2.4 kgenocorr_sp_1d()

```
real(sp) function mo_errormeasures::kgenocorr::kgenocorr_sp_1d (
             real(sp), dimension(:), intent(in) x,
             real(sp), dimension(:), intent(in) y,
             logical, dimension(:), intent(in), optional mask )
```

17.47.2.5 kgenocorr_sp_2d()

```
real(sp) function mo_errormeasures::kgenocorr::kgenocorr_sp_2d (
             real(sp), dimension(:, :), intent(in) x,
             real(sp), dimension(:, :), intent(in) y,
             logical, dimension(:, :), intent(in), optional mask )
```

17.47.2.6 kgenocorr_sp_3d()

```
real(sp) function mo_errormeasures::kgenocorr::kgenocorr_sp_3d (
             real(sp), dimension(:, :, :), intent(in) x,
             real(sp), dimension(:, :, :), intent(in) y,
             logical, dimension(:, :, :), intent(in), optional mask )
```

The documentation for this interface was generated from the following file:

- mo_errormeasures.f90

17.48 mo_moment::kurtosis Interface Reference

Public Member Functions

- real(sp) function kurtosis_sp (dat, mask)
- real(dp) function kurtosis_dp (dat, mask)

17.48.1 Member Function/Subroutine Documentation

17.48.1.1 kurtosis_dp()

```
real(dp) function mo_moment::kurtosis::kurtosis_dp (
             real(dp), dimension(:), intent(in) dat,
             logical, dimension(:), intent(in), optional mask )
```

17.48.1.2 kurtosis_sp()

```
real(sp) function mo_moment::kurtosis::kurtosis_sp (
             real(sp), dimension(:), intent(in) dat,
             logical, dimension(:), intent(in), optional mask )
```

The documentation for this interface was generated from the following file:

- mo_moment.f90

17.49 mo_utils::le Interface Reference

Public Member Functions

- elemental pure logical function lesserequal_sp (a, b)
- elemental pure logical function lesserequal_dp (a, b)

17.49.1 Member Function/Subroutine Documentation

17.49.1.1 lesserequal_dp()

```
elemental pure logical function mo_utils::le::lesserequal_dp (
             real(dp), intent(in) a,
             real(dp), intent(in) b )
```

17.49.1.2 lesserequal_sp()

```
elemental pure logical function mo_utils::le::lesserequal_sp (
             real(sp), intent(in) a,
             real(sp), intent(in) b )
```

The documentation for this interface was generated from the following file:

- mo_utils.f90

17.50 mo_utils::lesserequal Interface Reference

Public Member Functions

- elemental pure logical function lesserequal_sp (a, b)
- elemental pure logical function lesserequal_dp (a, b)

17.50.1 Member Function/Subroutine Documentation

17.50.1.1 lesserequal_dp()

```
elemental pure logical function mo_utils::lesserequal::lesserequal_dp (
             real(dp), intent(in) a,
             real(dp), intent(in) b )
```

17.50.1.2 lesserequal_sp()

```
elemental pure logical function mo_utils::lesserequal::lesserequal_sp (
             real(sp), intent(in) a,
             real(sp), intent(in) b )
```

The documentation for this interface was generated from the following file:

- mo_utils.f90

17.51 mo_linfit::linfit Interface Reference

Fits a straight line to input data by minimizing chi^2.

Public Member Functions

- real(sp) function, dimension(:), allocatable linfit_sp (x, y, a, b, siga, sigb, chi2, model2)
- real(dp) function, dimension(:), allocatable linfit_dp (x, y, a, b, siga, sigb, chi2, model2)

17.51.1 Detailed Description

Fits a straight line to input data by minimizing chi^2.

Given a set of data points x(1:ndata), y(1:ndata), fit them to a straight line $y = a + bx$ by minimizing chi2.
Model I minimizes y vs. x while Model II takes the geometric mean of y vs. x and x vs. y. Returned is the fitted line at x. Optional returns are a, b and their respective probable uncertainties siga and sigb, and the chi-square chi2.

Parameters

in	*real(sp/dp) :: x(:)*	1D-array with input x
in	*real(sp/dp) :: y(:)*	1D-array with input y
in	*logical, optional :: model2*	If present, use geometric mean regression instead of ordinary least square
out	*real(sp/dp), dimension(M) :: a*	intercept

Parameters

out	*real(sp/dp), dimension(M) :: b*	slope
out	*real(sp/dp), dimension(M) :: siga*	error on intercept
out	*real(sp/dp), dimension(M) :: sigb*	error on slope
out	*real(sp/dp) :: chisq*	Minimum chi^2

Returns

real(sp/dp), dimension(:), allocatable :: out — fitted values at x(:).

17.51.2 Member Function/Subroutine Documentation

17.51.2.1 linfit_dp()

```
real(dp) function, dimension(:), allocatable mo_linfit::linfit::linfit_dp (
             real(dp), dimension(:), intent(in) x,
             real(dp), dimension(:), intent(in) y,
             real(dp), intent(out), optional a,
             real(dp), intent(out), optional b,
             real(dp), intent(out), optional siga,
             real(dp), intent(out), optional sigb,
             real(dp), intent(out), optional chi2,
             logical, intent(in), optional model2 )
```

17.51.2.2 linfit_sp()

```
real(sp) function, dimension(:), allocatable mo_linfit::linfit::linfit_sp (
             real(sp), dimension(:), intent(in) x,
             real(sp), dimension(:), intent(in) y,
             real(sp), intent(out), optional a,
             real(sp), intent(out), optional b,
             real(sp), intent(out), optional siga,
             real(sp), intent(out), optional sigb,
             real(sp), intent(out), optional chi2,
             logical, intent(in), optional model2 )
```

The documentation for this interface was generated from the following file:

- mo_linfit.f90

17.52 mo_errormeasures::lnnse Interface Reference

Public Member Functions

- real(sp) function lnnse_sp_1d (x, y, mask)
- real(dp) function lnnse_dp_1d (x, y, mask)
- real(dp) function lnnse_dp_2d (x, y, mask)
- real(sp) function lnnse_sp_2d (x, y, mask)
- real(sp) function lnnse_sp_3d (x, y, mask)
- real(dp) function lnnse_dp_3d (x, y, mask)

17.52.1 Member Function/Subroutine Documentation

17.52.1.1 lnnse_dp_1d()

```
real(dp) function mo_errormeasures::lnnse::lnnse_dp_1d (
             real(dp), dimension(:), intent(in) x,
             real(dp), dimension(:), intent(in) y,
             logical, dimension(:), intent(inout), optional mask )
```

17.52.1.2 lnnse_dp_2d()

```
real(dp) function mo_errormeasures::lnnse::lnnse_dp_2d (
             real(dp), dimension(:, :), intent(in) x,
             real(dp), dimension(:, :), intent(in) y,
             logical, dimension(:, :), intent(inout), optional mask )
```

17.52.1.3 lnnse_dp_3d()

```
real(dp) function mo_errormeasures::lnnse::lnnse_dp_3d (
             real(dp), dimension(:, :, :), intent(in) x,
             real(dp), dimension(:, :, :), intent(in) y,
             logical, dimension(:, :, :), intent(inout), optional mask )
```

17.52.1.4 lnnse_sp_1d()

```
real(sp) function mo_errormeasures::lnnse::lnnse_sp_1d (
             real(sp), dimension(:), intent(in) x,
             real(sp), dimension(:), intent(in) y,
             logical, dimension(:), intent(inout), optional mask )
```

17.52.1.5 lnnse_sp_2d()

```
real(sp) function mo_errormeasures::lnnse::lnnse_sp_2d (
             real(sp), dimension(:, :), intent(in) x,
             real(sp), dimension(:, :), intent(in) y,
             logical, dimension(:, :), intent(inout), optional mask )
```

17.52.1.6 lnnse_sp_3d()

```
real(sp) function mo_errormeasures::lnnse::lnnse_sp_3d (
             real(sp), dimension(:, :, :), intent(in) x,
             real(sp), dimension(:, :, :), intent(in) y,
             logical, dimension(:, :, :), intent(inout), optional mask )
```

The documentation for this interface was generated from the following file:

- mo_errormeasures.f90

17.53 mo_utils::locate Interface Reference

Find closest values in a monotonic series, returns the indexes.

Public Member Functions

- integer(i4) function locate_0d_dp (x, y)
- integer(i4) function locate_0d_sp (x, y)
- integer(i4) function, dimension(:), allocatable locate_1d_dp (x, y)
- integer(i4) function, dimension(:), allocatable locate_1d_sp (x, y)

17.53.1 Detailed Description

Find closest values in a monotonic series, returns the indexes.

Given an array x(1:n), and given a value y, returns a value j such that y is between x(j) and x(j+1).
x must be monotonically increasing.
j=0 or j=N is returned to indicate that x is out of range.

Parameters

in	*real(dp/sp) :: x(:)*	Sorted array
in	*real(dp/sp) :: y[(:)]*	Value(s) of which the closest match in x(:) is wanted

Returns

integer(i4) :: index[(:)] — index(es) of x so that y is between x(index) and x(index+1)

Note

x must be monotonically increasing.

Author

Matthias Cuntz

Date

May 2014

17.53.2 Member Function/Subroutine Documentation

17.53.2.1 locate_0d_dp()

```
integer(i4) function mo_utils::locate::locate_0d_dp (
             real(dp), dimension(:), intent(in) x,
             real(dp), intent(in) y )
```

17.53.2.2 locate_0d_sp()

```
integer(i4) function mo_utils::locate::locate_0d_sp (
             real(sp), dimension(:), intent(in) x,
             real(sp), intent(in) y )
```

17.53.2.3 locate_1d_dp()

```
integer(i4) function, dimension(:), allocatable mo_utils::locate::locate_1d_dp (
             real(dp), dimension(:), intent(in) x,
             real(dp), dimension(:), intent(in) y )
```

17.53.2.4 locate_1d_sp()

```
integer(i4) function, dimension(:), allocatable mo_utils::locate::locate_1d_sp (
             real(sp), dimension(:), intent(in) x,
             real(sp), dimension(:), intent(in) y )
```

The documentation for this interface was generated from the following file:

- mo_utils.f90

17.54 mo_mad::mad Interface Reference

Public Member Functions

- logical function, dimension(size(arr)) mad_sp (arr, z, mask, deriv)
- logical function, dimension(size(arr)) mad_dp (arr, z, mask, deriv)
- real(dp) function, dimension(size(arr)) mad_val_dp (arr, z, mask, tout, mval)
- real(sp) function, dimension(size(arr)) mad_val_sp (arr, z, mask, tout, mval)

17.54.1 Member Function/Subroutine Documentation

17.54.1.1 mad_dp()

```
logical function, dimension(size(arr)) mo_mad::mad::mad_dp (
             real(dp), dimension(:), intent(in) arr,
             real(dp), intent(in), optional z,
             logical, dimension(:), intent(in), optional mask,
             integer(i4), intent(in), optional deriv )
```

17.54.1.2 mad_sp()

```
logical function, dimension(size(arr)) mo_mad::mad::mad_sp (
             real(sp), dimension(:), intent(in) arr,
             real(sp), intent(in), optional z,
```

```
            logical, dimension(:), intent(in), optional mask,
            integer(i4), intent(in), optional deriv )
```

17.54.1.3 mad_val_dp()

```
real(dp) function, dimension(size(arr)) mo_mad::mad::mad_val_dp (
            real(dp), dimension(:), intent(in) arr,
            real(dp), intent(in), optional z,
            logical, dimension(:), intent(in), optional mask,
            character(1) tout,
            real(dp), intent(in), optional mval )
```

17.54.1.4 mad_val_sp()

```
real(sp) function, dimension(size(arr)) mo_mad::mad::mad_val_sp (
            real(sp), dimension(:), intent(in) arr,
            real(sp), intent(in), optional z,
            logical, dimension(:), intent(in), optional mask,
            character(1) tout,
            real(sp), intent(in), optional mval )
```

The documentation for this interface was generated from the following file:

- mo_mad.f90

17.55 mo_errormeasures::mae Interface Reference

Public Member Functions

- real(sp) function mae_sp_1d (x, y, mask)
- real(dp) function mae_dp_1d (x, y, mask)
- real(sp) function mae_sp_2d (x, y, mask)
- real(dp) function mae_dp_2d (x, y, mask)
- real(sp) function mae_sp_3d (x, y, mask)
- real(dp) function mae_dp_3d (x, y, mask)

17.55.1 Member Function/Subroutine Documentation

17.55.1.1 mae_dp_1d()

```
real(dp) function mo_errormeasures::mae::mae_dp_1d (
            real(dp), dimension(:), intent(in) x,
            real(dp), dimension(:), intent(in) y,
            logical, dimension(:), intent(in), optional mask )
```

17.55.1.2 mae_dp_2d()

```
real(dp) function mo_errormeasures::mae::mae_dp_2d (
             real(dp), dimension(:, :), intent(in) x,
             real(dp), dimension(:, :), intent(in) y,
             logical, dimension(:, :), intent(in), optional mask )
```

17.55.1.3 mae_dp_3d()

```
real(dp) function mo_errormeasures::mae::mae_dp_3d (
             real(dp), dimension(:, :, :), intent(in) x,
             real(dp), dimension(:, :, :), intent(in) y,
             logical, dimension(:, :, :), intent(in), optional mask )
```

17.55.1.4 mae_sp_1d()

```
real(sp) function mo_errormeasures::mae::mae_sp_1d (
             real(sp), dimension(:), intent(in) x,
             real(sp), dimension(:), intent(in) y,
             logical, dimension(:), intent(in), optional mask )
```

17.55.1.5 mae_sp_2d()

```
real(sp) function mo_errormeasures::mae::mae_sp_2d (
             real(sp), dimension(:, :), intent(in) x,
             real(sp), dimension(:, :), intent(in) y,
             logical, dimension(:, :), intent(in), optional mask )
```

17.55.1.6 mae_sp_3d()

```
real(sp) function mo_errormeasures::mae::mae_sp_3d (
             real(sp), dimension(:, :, :), intent(in) x,
             real(sp), dimension(:, :, :), intent(in) y,
             logical, dimension(:, :, :), intent(in), optional mask )
```

The documentation for this interface was generated from the following file:

- mo_errormeasures.f90

17.56 mo_mcmc::mcmc Interface Reference

This module is Monte Carlo Markov Chain sampling of a posterior parameter distribution (measurement errors are either known or modeled).

Public Member Functions

- subroutine mcmc_dp (eval, likelihood, para, rangePar, mcmc_paras, burnin_paras, seed_in, printflag_↩ in, maskpara_in, restart, restart_file, tmp_file, loglike_in, ParaSelectMode_in, iter_burnin_in, iter_mcmc_in,

chains_in, stepsize_in)

17.56.1 Detailed Description

This module is Monte Carlo Markov Chain sampling of a posterior parameter distribution (measurement errors are either known or modeled).

Sample posterior parameter distribution with Metropolis Algorithm.
This sampling is performed in two steps, i.e. the burn-in phase for adjusting model dependent parameters for the second step which is the proper sampling using the Metropolis Hastings Algorithm.

This sampler does not change the best parameter set, i.e. it cannot be used as an optimiser.
However, the serial and the parallel version give therefore the bitwise same results. **1. BURN IN PHASE: FIND THE OPTIMAL STEP SIZE**

Purpose:
Find optimal stepsize for each parameter such that the acceptance ratio converges to a value around 0.3.
Important variables:

Variable	Description
burnin_iter	length of markov chain performed to calculate acceptance ratio
acceptance ratio	ratio between accepted jumps and all trials (LEN)
acceptance multiplier	stepsize of a parameter is multiplied with this value when jump is

accepted (initial : 1.01) rejection multiplier | stepsize of a parameter is multiplied with this value when jump is rejected (initial : 0.99 and will never be changed) stepsize | a new parameter value is chosen based on a uniform distribution pnew_i = pold_i + Unif(-stepsize_i, stepsize_i) (initial : stepsize_i = 1.0 for all i)
Algorithm:

1. start a new markov chain of length burnin_iter with initial parameter set is the OPTIMAL one
   - select a set of parameters to change:
     - accurate –> choose one parameter,
     - comput. efficient –> choose all parameters,
     - moderate accurate & efficient –> choose half of the parameters
   - change parameter(s) based on their stepsize
   - decide whether changed parameter set is accepted or rejected:
     - $\text{oddsRatio} = \frac{\text{likelihood}(p_{new})}{\text{likelihood}(p_{old})}$
     - random number r = Uniform[0,1]
     - odds Ratio $>$ 0 –> positive accept
       odds Ratio $>$ r –> negative accept
       odds Ratio $<$ r –> reject

- adapt stepsize of parameters changed:
  - accepted step: stepsize_i = stepsize_i $*$ accptance multiplier
  - rejected step: stepsize_i = stepsize_i $*$ rejection multiplier
- if step is accepted: for all changed parameter(s) change stepsize

2. calculate acceptance ratio of the Markov Chain

3. adjust acceptance multiplier acc_mult and store good ratios in history list

   - acceptance ratio $<$ 0.23 –$>$ acc_mult = acc_mult $*$ 0.99
     delete history list
   - acceptance ratio $>$ 0.44 –$>$ acc_mult = acc_mult $*$ 1.01
     delete history list
   - 0.23 $<$ acceptance ratio $<$ 0.44
     add acceptance ratio to history list

4. check if already 10 values are stored in history list and if they have converged to a value above 0.3
   (mean above 0.3 and variance less $\sqrt{1/12 * 0.05^2}$ = Variance of uniform [acc_ratio +/- 2.5%])

   - if check is positive abort and save stepsizes
     else goto (1)

2. MONTE CARLO MARKOV CHAIN: SAMPLE POSTERIOR DISTRIBUTION OF PARAMETER

Purpose:
use the previous adapted stepsizes and perform ONE monte carlo markov chain
the accepted parameter sets show the posterior distribution of parameters
Important variables:

Variable	Description
iter_mcmc	length of the markov chain ($>>$ iter_burnin)
stepsize	a new parameter value is chosen based on a uniform distribution

pnew_i = pold_i + Unif(-stepsize_i, stepsize_i) use stepsizes of the burn-in (1)

Algorithm:

1. select a set of parameters to change

   - accurate –$>$ choose one parameter,

- comput. efficient –> choose all parameters,
- moderate accurate & efficient –> choose half of the parameters

2. change parameter(s) based on their stepsize
3. decide whether changed parameter set is accepted or rejected:
   - $\text{oddsRatio} = \frac{\text{likelihood}(p_{new})}{\text{likelihood}(p_{old})}$
   - random number r = Uniform[0,1]
   - odds Ratio $>$ 0 –> positive accept
     odds Ratio $>$ r –> negative accept
     odds Ratio $<$ r –> reject
4. if step is accepted: save parameter set
5. goto (1)

Parameters

in	*real(dp) :: likelihood(x)*	Interface Function which calculates likelihood of given parameter set x
in	*real(dp) :: para(:)*	Inital parameter set (should be GOOD approximation of best parameter set)
in	*real(dp) :: rangePar(size(para),2)*	Min/max range of parameters
out	*real(dp), allocatable :: mcmc_paras(:,:)*	Parameter sets sampled in proper MCMC part of algorithm
out	*real(dp), allocatable :: burnin_paras(:,:)*	Parameter sets sampled during burn-in part of algorithm
in	*integer(i8), optional :: seed_in*	User seed to initialise the random number generator (default: none –> initialized with timeseed)
in	*logical, optional :: printflag_in*	Print of output on command line (default: .False.)
in	*logical, optional :: maskpara_in(size(para))*	Parameter will be sampled (.True.) or not (.False.) (default: .True.)
in	*character(len=*), optional :: tmp_file*	filename for temporal data saving: every iter_mcmc_in iterations parameter sets are appended to this file the number of the chain will be prepended to filename output format: netcdf (default: no file writing)
in	*logical, optional :: loglike_in*	true if loglikelihood function is given instead of likelihood function (default: .false.)

Parameters

in	*integer(i4), optional :: ParaSelectMode_in*	How many parameters will be changed at once? • half of the parameter –> 1_i4 • only one parameter –> 2_i4 • all parameter –> 3_i4 (default: 2_i4)
in	*integer(i4), optional :: iter_burnin_in*	Length of Markov chains of initial burn-in part (default: Max(250, 200∗count(maskpara)))
in	*integer(i4), optional :: iter_mcmc_in*	Length of Markov chains of proper MCMC part (default: 1000 ∗ count(maskpara))
in	*integer(i4), optional :: chains_in*	number of parallel mcmc chains (default: 5_i4)
in	*real(dp), DIMENSION(size(para,1)), optional :: stepsize_in*	stepsize for each parameter if given burn-in is discarded (default: none –> adjusted in burn-in)

Note

Likelihood has to be defined as a function interface
The maximal number of parameters is 1000.

17.56.2 Member Function/Subroutine Documentation

17.56.2.1 mcmc_dp()

```
subroutine mo_mcmc::mcmc::mcmc_dp (
             procedure(eval_interface), intent(in), pointer eval,
             procedure(objective_interface), intent(in), pointer likelihood,
             real(dp), dimension(:), intent(in) para,
             real(dp), dimension(:, :), intent(in) rangePar,
             real(dp), dimension(:, :), intent(out), allocatable mcmc_paras,
             real(dp), dimension(:, :), intent(out), allocatable burnin_paras,
             integer(i8), intent(in), optional seed_in,
             logical, intent(in), optional printflag_in,
             logical, dimension(size(para, 1)), intent(in), optional maskpara_in,
             logical, intent(in), optional restart,
             character(len = *), intent(in), optional restart_file,
             character(len = *), intent(in), optional tmp_file,
             logical, intent(in), optional loglike_in,
             integer(i4), intent(in), optional ParaSelectMode_in,
             integer(i4), intent(in), optional iter_burnin_in,
             integer(i4), intent(in), optional iter_mcmc_in,
             integer(i4), intent(in), optional chains_in,
             real(dp), dimension(size(para, 1)), intent(in), optional stepsize_in )
```

The documentation for this interface was generated from the following file:

- mo_mcmc.f90

17.57 mo_mcmc::mcmc_stddev Interface Reference

This module is Monte Carlo Markov Chain sampling of a posterior parameter distribution (measurement errors are neither known nor modeled).

Public Member Functions

- subroutine mcmc_stddev_dp (eval, likelihood, para, rangePar, mcmc_paras, burnin_paras, seed_in, printflag_in, maskpara_in, tmp_file, loglike_in, ParaSelectMode_in, iter_burnin_in, iter_mcmc_in, chains_in, stepsize_in)

17.57.1 Detailed Description

This module is Monte Carlo Markov Chain sampling of a posterior parameter distribution (measurement errors are neither known nor modeled).

Sample posterior parameter distribution with Metropolis Algorithm.
This sampling is performed in two steps, i.e. the burn-in phase for adjusting model dependent parameters for the second step which is the proper sampling using the Metropolis Hastings Algorithm.

1. BURN IN PHASE: FIND THE OPTIMAL STEP SIZE

Purpose:
Find optimal stepsize for each parameter such that the acceptance ratio converges to a value around 0.3.
Important variables:

Variable	Description
burnin_iter	length of markov chain performed to calculate acceptance ratio
acceptance ratio	ratio between accepted jumps and all trials (LEN)
acceptance multiplier	stepsize of a parameter is multiplied with this value when jump is

accepted (initial : 1.01) rejection multiplier | stepsize of a parameter is multiplied with this value when jump is rejected (initial : 0.99 and will never be changed) stepsize | a new parameter value is chosen based on a uniform distribution pnew_i = pold_i + Unif(-stepsize_i, stepsize_i) (initial : stepsize_i = 1.0 for all i)
Algorithm:

1. start a new markov chain of length burnin_iter with initial parameter set is the OPTIMAL one
    - select a set of parameters to change:
        - accurate –> choose one parameter,
        - comput. efficient –> choose all parameters,
        - moderate accurate & efficient –> choose half of the parameters
    - change parameter(s) based on their stepsize
    - decide whether changed parameter set is accepted or rejected:

  - $\text{oddsRatio} = \frac{\text{likelihood}(p_{new})}{\text{likelihood}(p_{old})}$
  - random number r = Uniform[0,1]
  - odds Ratio $>$ 0 –$>$ positive accept
    odds Ratio $>$ r –$>$ negative accept
    odds Ratio $<$ r –$>$ reject

- adapt stepsize of parameters changed:
  - accepted step: stepsize_i = stepsize_i $*$ accptance multiplier
  - rejected step: stepsize_i = stepsize_i $*$ rejection multiplier
- if step is accepted: for all changed parameter(s) change stepsize

2. calculate acceptance ratio of the Markov Chain
3. adjust acceptance multiplier acc_mult and store good ratios in history list
   - acceptance ratio $<$ 0.23 –$>$ acc_mult = acc_mult $*$ 0.99
     delete history list
   - acceptance ratio $>$ 0.44 –$>$ acc_mult = acc_mult $*$ 1.01
     delete history list
   - 0.23 $<$ acceptance ratio $<$ 0.44
     add acceptance ratio to history list
4. check if already 10 values are stored in history list and if they have converged to a value above 0.3
   (mean above 0.3 and variance less $\sqrt{1/12 * 0.05^2}$ = Variance of uniform [acc_ratio +/- 2.5%])
   - if check is positive abort and save stepsizes
     else goto (1)

2. MONTE CARLO MARKOV CHAIN: SAMPLE POSTERIOR DISTRIBUTION OF PARAMETER

Purpose:
use the previous adapted stepsizes and perform ONE monte carlo markov chain
the accepted parameter sets show the posterior distribution of parameters
Important variables:

Variable	Description
iter_mcmc	length of the markov chain ($>>$ iter_burnin)
stepsize	a new parameter value is chosen based on a uniform distribution

pnew_i = pold_i + Unif(-stepsize_i, stepsize_i) use stepsizes of the burn-in (1)

Algorithm:

1. select a set of parameters to change
   - accurate –> choose one parameter,
   - comput. efficient –> choose all parameters,
   - moderate accurate & efficient –> choose half of the parameters
2. change parameter(s) based on their stepsize
3. decide whether changed parameter set is accepted or rejected:
   - $\text{oddsRatio} = \frac{\text{likelihood}(p_{new})}{\text{likelihood}(p_{old})}$
   - random number r = Uniform[0,1]
   - odds Ratio $>$ 0 –> positive accept
     odds Ratio $>$ r –> negative accept
     odds Ratio $<$ r –> reject
4. if step is accepted: save parameter set
5. goto (1)

Parameters

in	*real(dp) :: likelihood(x,sigma,stddev_new,likeli_new)*	Interface Function which calculates likelihood of given parameter set x and given standard deviation sigma and returns optionally the standard deviation stddev_new of the errors using x and likelihood likeli_new using stddev_new
in	*real(dp) :: para(:)*	Inital parameter set (should be GOOD approximation of best parameter set)
in	*real(dp) :: rangePar(size(para),2)*	Min/max range of parameters
out	*real(dp), allocatable :: mcmc_paras(:,:)*	Parameter sets sampled in proper MCMC part of algorithm
out	*real(dp), allocatable :: burnin_paras(:,:)*	Parameter sets sampled during burn-in part of algorithm
in	*integer(i8), optional :: seed_in*	User seed to initialise the random number generator (default: none –> initialized with timeseed)
in	*logical, optional :: printflag_in*	Print of output on command line (default: .False.)
in	*logical, optional :: maskpara_in(size(para))*	Parameter will be sampled (.True.) or not (.False.) (default: .True.)

Parameters

in	*character(len=∗), optional :: tmp_file*	filename for temporal data saving: every iter_mcmc_in iterations parameter sets are appended to this file the number of the chain will be prepended to filename output format: netcdf (default: no file writing)
in	*logical, optional :: loglike_in*	true if loglikelihood function is given instead of likelihood function (default: .false.)
in	*integer(i4), optional :: ParaSelectMode_in*	How many parameters will be changed at once? • half of the parameter –> 1_i4 • only one parameter –> 2_i4 • all parameter –> 3_i4 (default: 2_i4)
in	*integer(i4), optional :: iter_burnin_in*	Length of Markov chains of initial burn-in part (default: Max(250, 200∗count(maskpara)))
in	*integer(i4), optional :: iter_mcmc_in*	Length of Markov chains of proper MCMC part (default: 1000 ∗ count(maskpara))
in	*integer(i4), optional :: chains_in*	number of parallel mcmc chains (default: 5_i4)
in	*real(dp), DIMENSION(size(para,1)), optional :: stepsize_in*	stepsize for each parameter if given burn-in is discarded (default: none –> adjusted in burn-in)

17.57.2 Member Function/Subroutine Documentation

17.57.2.1 mcmc_stddev_dp()

```
subroutine mo_mcmc::mcmc_stddev::mcmc_stddev_dp (
             procedure(eval_interface), intent(in), pointer eval,
             procedure(objective_interface), intent(in), pointer likelihood,
             real(dp), dimension(:), intent(in) para,
             real(dp), dimension(:, :), intent(in) rangePar,
             real(dp), dimension(:, :), intent(out), allocatable mcmc_paras,
             real(dp), dimension(:, :), intent(out), allocatable burnin_paras,
             integer(i8), intent(in), optional seed_in,
             logical, intent(in), optional printflag_in,
             logical, dimension(size(para, 1)), intent(in), optional maskpara_in,
             character(len = *), intent(in), optional tmp_file,
             logical, intent(in), optional loglike_in,
             integer(i4), intent(in), optional ParaSelectMode_in,
             integer(i4), intent(in), optional iter_burnin_in,
             integer(i4), intent(in), optional iter_mcmc_in,
             integer(i4), intent(in), optional chains_in,
             real(dp), dimension(size(para, 1)), intent(in), optional stepsize_in )
```

The documentation for this interface was generated from the following file:

- mo_mcmc.f90

17.58 mo_moment::mean Interface Reference

Public Member Functions

- real(sp) function mean_sp (dat, mask)
- real(dp) function mean_dp (dat, mask)

17.58.1 Member Function/Subroutine Documentation

17.58.1.1 mean_dp()

```
real(dp) function mo_moment::mean::mean_dp (
             real(dp), dimension(:), intent(in) dat,
             logical, dimension(:), intent(in), optional mask )
```

17.58.1.2 mean_sp()

```
real(sp) function mo_moment::mean::mean_sp (
             real(sp), dimension(:), intent(in) dat,
             logical, dimension(:), intent(in), optional mask )
```

The documentation for this interface was generated from the following file:

- mo_moment.f90

17.59 mo_template::mean Interface Reference

The average.

Public Member Functions

- real(sp) function mean_sp (dat, mask)
- real(dp) function mean_dp (dat, mask)

17.59.1 Detailed Description

The average.

Calculates the average value of a vector, i.e. the first moment of a series of numbers:

$$\bar{x} = \frac{1}{N} \sum_{i=1}^{N} x_i$$

If an optinal mask is given, the mean is only over those locations that correspond to true values in the mask. x can be single or double precision. The result will have the same numerical precision. ADDITIONAL INFORMATION vec = (/ 1., 2, 3., -999., 5., 6. /) m = mean(vec, mask=(vec >= 0.)) -> see also example in test directory Sokal

RR & Rohlf FJ - Biometry: the principle and practice of statistics in biological research, Freeman & Co., ISBN 0-7167-2411-1 Press WH, Teukolsky SA, Vetterling WT, & Flannery BP - Numerical Recipes in Fortran 90 - The Art of Parallel Scientific Computing, 2nd Edition, Volume 2 of Fortran Numerical Recipes, Cambridge University Press, UK, 1996

Returns

real(sp/dp) :: mean — $\bar{x}$ average of all elements in vec

Authors

Matthias Cuntz

Date

Nov 2011

17.59.2 Member Function/Subroutine Documentation

17.59.2.1 mean_dp()

```
real(dp) function mo_template::mean::mean_dp (
             real(dp), dimension(:), intent(in) dat,
             logical, dimension(:), intent(in), optional mask )
```

17.59.2.2 mean_sp()

```
real(sp) function mo_template::mean::mean_sp (
             real(sp), dimension(:), intent(in) dat,
             logical, dimension(:), intent(in), optional mask )
```

The documentation for this interface was generated from the following file:

- mo_template.f90

17.60 mo_percentile::median Interface Reference

Public Member Functions

- real(sp) function median_sp (arrin, mask)
- real(dp) function median_dp (arrin, mask)

17.60.1 Member Function/Subroutine Documentation

17.60.1.1 median_dp()

```
real(dp) function mo_percentile::median::median_dp (
             real(dp), dimension(:), intent(in) arrin,
             logical, dimension(:), intent(in), optional mask )
```

17.60.1.2 median_sp()

```
real(sp) function mo_percentile::median::median_sp (
             real(sp), dimension(:), intent(in) arrin,
             logical, dimension(:), intent(in), optional mask )
```

The documentation for this interface was generated from the following file:

- mo_percentile.f90

17.61 mo_moment::mixed_central_moment Interface Reference

Public Member Functions

- real(sp) function mixed_central_moment_sp (x, y, r, s, mask)
- real(dp) function mixed_central_moment_dp (x, y, r, s, mask)

17.61.1 Member Function/Subroutine Documentation

17.61.1.1 mixed_central_moment_dp()

```
real(dp) function mo_moment::mixed_central_moment::mixed_central_moment_dp (
             real(dp), dimension(:), intent(in) x,
             real(dp), dimension(:), intent(in) y,
             integer(i4), intent(in) r,
             integer(i4), intent(in) s,
             logical, dimension(:), intent(in), optional mask )
```

17.61.1.2 mixed_central_moment_sp()

```
real(sp) function mo_moment::mixed_central_moment::mixed_central_moment_sp (
             real(sp), dimension(:), intent(in) x,
             real(sp), dimension(:), intent(in) y,
             integer(i4), intent(in) r,
             integer(i4), intent(in) s,
             logical, dimension(:), intent(in), optional mask )
```

The documentation for this interface was generated from the following file:

- mo_moment.f90

17.62 mo_moment::mixed_central_moment_var Interface Reference

Public Member Functions

- real(sp) function mixed_central_moment_var_sp (x, y, r, s, mask)
- real(dp) function mixed_central_moment_var_dp (x, y, r, s, mask)

17.62.1 Member Function/Subroutine Documentation

17.62.1.1 mixed_central_moment_var_dp()

```
real(dp) function mo_moment::mixed_central_moment_var::mixed_central_moment_var_dp (
             real(dp), dimension(:), intent(in) x,
             real(dp), dimension(:), intent(in) y,
             integer(i4), intent(in) r,
             integer(i4), intent(in) s,
             logical, dimension(:), intent(in), optional mask )
```

17.62.1.2 mixed_central_moment_var_sp()

```
real(sp) function mo_moment::mixed_central_moment_var::mixed_central_moment_var_sp (
             real(sp), dimension(:), intent(in) x,
             real(sp), dimension(:), intent(in) y,
             integer(i4), intent(in) r,
             integer(i4), intent(in) s,
             logical, dimension(:), intent(in), optional mask )
```

The documentation for this interface was generated from the following file:

- mo_moment.f90

17.63 mo_moment::moment Interface Reference

Public Member Functions

- subroutine moment_sp (dat, average, variance, skewness, kurtosis, mean, stddev, absdev, mask)
- subroutine moment_dp (dat, average, variance, skewness, kurtosis, mean, stddev, absdev, mask)

17.63.1 Member Function/Subroutine Documentation

17.63.1.1 moment_dp()

```
subroutine mo_moment::moment::moment_dp (
             real(dp), dimension(:), intent(in) dat,
             real(dp), intent(out), optional average,
             real(dp), intent(out), optional variance,
             real(dp), intent(out), optional skewness,
```

```
            real(dp), intent(out), optional kurtosis,
            real(dp), intent(out), optional mean,
            real(dp), intent(out), optional stddev,
            real(dp), intent(out), optional absdev,
            logical, dimension(:), intent(in), optional mask )
```

17.63.1.2 moment_sp()

```
subroutine mo_moment::moment::moment_sp (
            real(sp), dimension(:), intent(in) dat,
            real(sp), intent(out), optional average,
            real(sp), intent(out), optional variance,
            real(sp), intent(out), optional skewness,
            real(sp), intent(out), optional kurtosis,
            real(sp), intent(out), optional mean,
            real(sp), intent(out), optional stddev,
            real(sp), intent(out), optional absdev,
            logical, dimension(:), intent(in), optional mask )
```

The documentation for this interface was generated from the following file:

- mo_moment.f90

17.64 mo_orderpack::mrgref Interface Reference

Public Member Functions

- subroutine d_mrgref (XVALT, IRNGT)
- subroutine r_mrgref (XVALT, IRNGT)
- subroutine i_mrgref (XVALT, IRNGT)

17.64.1 Member Function/Subroutine Documentation

17.64.1.1 d_mrgref()

```
subroutine mo_orderpack::mrgref::d_mrgref (
            real(kind = dp), dimension (:), intent(in) XVALT,
            integer(kind = i4), dimension (:), intent(out) IRNGT )
```

17.64.1.2 i_mrgref()

```
subroutine mo_orderpack::mrgref::i_mrgref (
            integer(kind = i4), dimension (:), intent(in) XVALT,
            integer(kind = i4), dimension (:), intent(out) IRNGT )
```

17.64.1.3 r_mrgref()

```
subroutine mo_orderpack::mrgref::r_mrgref (
             real(kind = sp), dimension (:), intent(in) XVALT,
             integer(kind = i4), dimension (:), intent(out) IRNGT )
```

The documentation for this interface was generated from the following file:

- mo_orderpack.f90

17.65 mo_orderpack::mrgrnk Interface Reference

Public Member Functions

- subroutine d_mrgrnk (XDONT, IRNGT)
- subroutine r_mrgrnk (XDONT, IRNGT)
- subroutine i_mrgrnk (XDONT, IRNGT)

17.65.1 Member Function/Subroutine Documentation

17.65.1.1 d_mrgrnk()

```
subroutine mo_orderpack::mrgrnk::d_mrgrnk (
             real(kind = dp), dimension (:), intent(in) XDONT,
             integer(kind = i4), dimension (:), intent(out) IRNGT )
```

17.65.1.2 i_mrgrnk()

```
subroutine mo_orderpack::mrgrnk::i_mrgrnk (
             integer(kind = i4), dimension (:), intent(in) XDONT,
             integer(kind = i4), dimension (:), intent(out) IRNGT )
```

17.65.1.3 r_mrgrnk()

```
subroutine mo_orderpack::mrgrnk::r_mrgrnk (
             real(kind = sp), dimension (:), intent(in) XDONT,
             integer(kind = i4), dimension (:), intent(out) IRNGT )
```

The documentation for this interface was generated from the following file:

- mo_orderpack.f90

17.66 mo_errormeasures::mse Interface Reference

Public Member Functions

- real(sp) function mse_sp_1d (x, y, mask)

- real(dp) function mse_dp_1d (x, y, mask)
- real(sp) function mse_sp_2d (x, y, mask)
- real(dp) function mse_dp_2d (x, y, mask)
- real(sp) function mse_sp_3d (x, y, mask)
- real(dp) function mse_dp_3d (x, y, mask)

17.66.1 Member Function/Subroutine Documentation

17.66.1.1 mse_dp_1d()

```
real(dp) function mo_errormeasures::mse::mse_dp_1d (
             real(dp), dimension(:), intent(in) x,
             real(dp), dimension(:), intent(in) y,
             logical, dimension(:), intent(in), optional mask )
```

17.66.1.2 mse_dp_2d()

```
real(dp) function mo_errormeasures::mse::mse_dp_2d (
             real(dp), dimension(:, :), intent(in) x,
             real(dp), dimension(:, :), intent(in) y,
             logical, dimension(:, :), intent(in), optional mask )
```

17.66.1.3 mse_dp_3d()

```
real(dp) function mo_errormeasures::mse::mse_dp_3d (
             real(dp), dimension(:, :, :), intent(in) x,
             real(dp), dimension(:, :, :), intent(in) y,
             logical, dimension(:, :, :), intent(in), optional mask )
```

17.66.1.4 mse_sp_1d()

```
real(sp) function mo_errormeasures::mse::mse_sp_1d (
             real(sp), dimension(:), intent(in) x,
             real(sp), dimension(:), intent(in) y,
             logical, dimension(:), intent(in), optional mask )
```

17.66.1.5 mse_sp_2d()

```
real(sp) function mo_errormeasures::mse::mse_sp_2d (
             real(sp), dimension(:, :), intent(in) x,
             real(sp), dimension(:, :), intent(in) y,
             logical, dimension(:, :), intent(in), optional mask )
```

17.66.1.6 mse_sp_3d()

```
real(sp) function mo_errormeasures::mse::mse_sp_3d (
             real(sp), dimension(:, :, :), intent(in) x,
             real(sp), dimension(:, :, :), intent(in) y,
             logical, dimension(:, :, :), intent(in), optional mask )
```

The documentation for this interface was generated from the following file:

- mo_errormeasures.f90

17.67 mo_orderpack::mulcnt Interface Reference

Public Member Functions

- subroutine d_mulcnt (XDONT, IMULT)
- subroutine r_mulcnt (XDONT, IMULT)
- subroutine i_mulcnt (XDONT, IMULT)

17.67.1 Member Function/Subroutine Documentation

17.67.1.1 d_mulcnt()

```
subroutine mo_orderpack::mulcnt::d_mulcnt (
             real(kind = dp), dimension (:), intent(in) XDONT,
             integer(kind = i4), dimension (:), intent(out) IMULT )
```

17.67.1.2 i_mulcnt()

```
subroutine mo_orderpack::mulcnt::i_mulcnt (
             integer(kind = i4), dimension (:), intent(in) XDONT,
             integer(kind = i4), dimension (:), intent(out) IMULT )
```

17.67.1.3 r_mulcnt()

```
subroutine mo_orderpack::mulcnt::r_mulcnt (
             real(kind = sp), dimension (:), intent(in) XDONT,
             integer(kind = i4), dimension (:), intent(out) IMULT )
```

The documentation for this interface was generated from the following file:

- mo_orderpack.f90

17.68 mo_percentile::n_element Interface Reference

Public Member Functions

- real(dp) function n_element_dp (idat, n, mask, before, after, previous, next)

- real(sp) function n_element_sp (idat, n, mask, before, after, previous, next)

17.68.1 Member Function/Subroutine Documentation

17.68.1.1 n_element_dp()

```
real(dp) function mo_percentile::n_element::n_element_dp (
             real(dp), dimension(:), intent(in) idat,
             integer(i4), intent(in) n,
             logical, dimension(:), intent(in), optional mask,
             real(dp), intent(out), optional before,
             real(dp), intent(out), optional after,
             real(dp), intent(out), optional previous,
             real(dp), intent(out), optional next )
```

17.68.1.2 n_element_sp()

```
real(sp) function mo_percentile::n_element::n_element_sp (
             real(sp), dimension(:), intent(in) idat,
             integer(i4), intent(in) n,
             logical, dimension(:), intent(in), optional mask,
             real(sp), intent(out), optional before,
             real(sp), intent(out), optional after,
             real(sp), intent(out), optional previous,
             real(sp), intent(out), optional next )
```

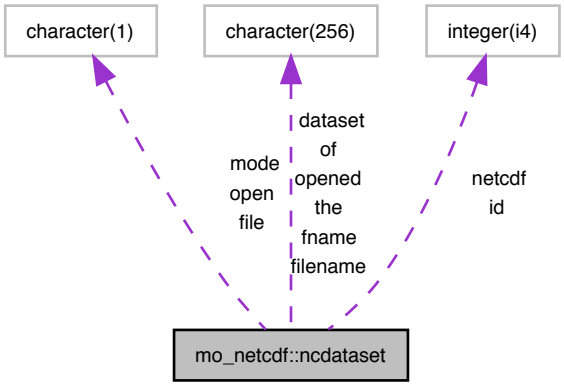
The documentation for this interface was generated from the following file:

- mo_percentile.f90

17.69 mo_netcdf::ncdataset Interface Reference

Provides basic file modification functionality.

Collaboration diagram for mo_netcdf::ncdataset:



Public Member Functions

- procedure, public initncdataset
- procedure, public getnovariables
- procedure, public getvariableids
- procedure, public getvariables
- procedure, public hasvariable

  Check if variable exists.
- procedure, public hasdimension

  Check if dimension exists.
- procedure, public getunlimiteddimension

  Return the unlimited dimension of the dataset.
- procedure, public isunlimited => isDatasetUnlimited

  Check if the dataset is unlimited.
- procedure, public close

  Close the datset.
- procedure, public setdimension

  Create a new dimension.
- generic, public setattribute => setGlobalAttributeChar, setGlobalAttributeI8, setGlobalAttributeI16, setGlobalAttributeI32, setGlobalAttributeI64, setGlobalAttributeF32, setGlobalAttributeF64

  Create a new global Attribute.
- generic, public getattribute => getGlobalAttributeChar, getGlobalAttributeI8, getGlobalAttributeI16, getGlobalAttributeI32, getGlobalAttributeI64, getGlobalAttributeF32, getGlobalAttributeF64

  Retrieve global attribute value.
- generic, public getdimension => getDimensionById, getDimensionByName

  Retrieve NcDimension.
- generic, public setvariable => setVariableWithNames, setVariableWithTypes, setVariableWithIds

  Create a NetCDF variable.
- generic, public getvariable => getVariableByName

  Retrieve NcVariable.

Public Attributes

- character(256) fname
- character(256) filename
- character(256) of
- character(256) the
- character(256) opened
- character(256) dataset
- character(1) mode
- character(1) file
- character(1) open
- integer(i4) id
- integer(i4) netcdf

Private Member Functions

- procedure, private setglobalattributechar
- procedure, private setglobalattributei8
- procedure, private setglobalattributei16
- procedure, private setglobalattributei32
- procedure, private setglobalattributei64
- procedure, private setglobalattributef32
- procedure, private setglobalattributef64
- procedure, private getglobalattributechar
- procedure, private getglobalattributei8
- procedure, private getglobalattributei16
- procedure, private getglobalattributei32
- procedure, private getglobalattributei64
- procedure, private getglobalattributef32
- procedure, private getglobalattributef64
- procedure, private getdimensionbyname
- procedure, private getdimensionbyid
- procedure, private setvariablewithtypes
- procedure, private setvariablewithnames
- procedure, private setvariablewithids
- procedure, private getvariablebyname

17.69.1 Detailed Description

Provides basic file modification functionality.

Bound to this derived type is the basic file level create/retrieve functionality, i.e. functions/subroutines to create/retrieve dimensions, variables and global attributes. All files created by this derived type and its procedures are are NF90_NETCDF4 only. The supported modes are: r: read w: write/create

Parameters

in	*character(∗) :: fname*	
in	*character(1) :: mode*	

Returns

"type(NcDataset)"

17.69.2 Member Function/Subroutine Documentation

17.69.2.1 close()

`procedure, public mo_netcdf::ncdataset::close ( )`

Close the datset.

Close the NetCDF datset. The program will teminate abruptly if the file cannot be closed correctly.

Author

David Schaefer

Date

June 2015

17.69.2.2 getattribute()

`generic, public mo_netcdf::ncdataset::getattribute ( )`

Retrieve global attribute value.

Retrieve the value for a global attribute specified by its name. The program will teminate abruptly if the attribute does not exist.

Parameters

in	*character(∗) :: name*	
out	*character(∗)/integer(i4)/real(sp)/real(dp) :: value*	

Author

David Schaefer

Date

June 2015

17.69.2.3 getdimension()

`generic, public mo_netcdf::ncdataset::getdimension ( )`

Retrieve NcDimension.

Retrieve the NcDimension derived type for the dimension specified by its name or id. The program will teminate abruptly if no such dimension exists.

Parameters

in	*character(∗)/integer(i4) :: name/id*	

Returns

NcDimension

Author

David Schaefer

Date

June 2015

17.69.2.4 getdimensionbyid()

```
procedure, private mo_netcdf::ncdataset::getdimensionbyid ( )  [private]
```

17.69.2.5 getdimensionbyname()

```
procedure, private mo_netcdf::ncdataset::getdimensionbyname ( )  [private]
```

17.69.2.6 getglobalattributechar()

```
procedure, private mo_netcdf::ncdataset::getglobalattributechar ( )  [private]
```

17.69.2.7 getglobalattributef32()

```
procedure, private mo_netcdf::ncdataset::getglobalattributef32 ( )  [private]
```

17.69.2.8 getglobalattributef64()

```
procedure, private mo_netcdf::ncdataset::getglobalattributef64 ( )  [private]
```

17.69.2.9 getglobalattributei16()

```
procedure, private mo_netcdf::ncdataset::getglobalattributei16 ( )  [private]
```

17.69.2.10 getglobalattributei32()

```
procedure, private mo_netcdf::ncdataset::getglobalattributei32 ( )  [private]
```

17.69.2.11 getglobalattributei64()

```
procedure, private mo_netcdf::ncdataset::getglobalattributei64 ( )  [private]
```

17.69.2.12 getglobalattributei8()

```
procedure, private mo_netcdf::ncdataset::getglobalattributei8 ( )  [private]
```

17.69.2.13 getnovariables()

```
procedure, public mo_netcdf::ncdataset::getnovariables ( )
```

17.69.2.14 getunlimiteddimension()

```
procedure, public mo_netcdf::ncdataset::getunlimiteddimension ( )
```

Return the unlimited dimension of the dataset.

Returns the NcDimension derived type of the unlimited dimension. The program will teminate abruptly if no such dimension exists.

Parameters

in	*character(∗) :: name*	

Returns

"logical"

Author

David Schaefer

Date

June 2015

17.69.2.15 getvariable()

```
generic, public mo_netcdf::ncdataset::getvariable ( )
```

Retrieve NcVariable.

Retrieve the NcVariable derived type for the variable specified by its name. The program will teminate abruptly if no such dimension exists.

Parameters

in	*character(∗) :: name*	

Returns

NcVariable

Author

David Schaefer

Date

June 2015

17.69.2.16 getvariablebyname()

```
procedure, private mo_netcdf::ncdataset::getvariablebyname ( )  [private]
```

17.69.2.17 getvariableids()

```
procedure, public mo_netcdf::ncdataset::getvariableids ( )
```

17.69.2.18 getvariables()

```
procedure, public mo_netcdf::ncdataset::getvariables ( )
```

17.69.2.19 hasdimension()

```
procedure, public mo_netcdf::ncdataset::hasdimension ( )
```

Check if dimension exists.

Returns true if a dimension with the given name exists, false otherwise.

Parameters

in	*character(∗) :: name*	

Returns

"logical"

Author

David Schaefer

Date

June 2015

17.69.2.20 hasvariable()

```
procedure, public mo_netcdf::ncdataset::hasvariable ( )
```

Check if variable exists.

Returns true if a variable with the given name exists, false otherwise.

Parameters

in	*character(∗) :: name*	

Returns

"logical"

Author

David Schaefer

Date

June 2015

17.69.2.21 initncdataset()

```
procedure, public mo_netcdf::ncdataset::initncdataset ( )
```

17.69.2.22 isunlimited()

```
procedure, public mo_netcdf::ncdataset::isunlimited ( )
```

Check if the dataset is unlimited.

Returns true if the dataset contains an unlimited dimension, false otherwise.

Returns

"logical"

Author

David Schaefer

Date

June 2015

17.69.2.23 setattribute()

```
generic, public mo_netcdf::ncdataset::setattribute ( )
```

Create a new global Attribute.

Create a new global attribute from given name and value. The program will teminate abruptly if the attribute cannot be created.

Parameters

in	*character(∗) :: name*	
in	*character(∗)/integer(i4)/real(sp)/real(dp) :: value*	

Author

David Schaefer

Date

June 2015

17.69.2.24 setdimension()

`procedure, public mo_netcdf::ncdataset::setdimension ( )`

Create a new dimension.

Create a new dimension of given length. A length < 0 indicates an unlimited dimension. The program will teminate abruptly if the dimension cannot be created.

Parameters

in	*character(∗) :: name*	
in	*integer(i4) :: length*	

Returns

NcDimension

Author

David Schaefer

Date

June 2015

17.69.2.25 setglobalattributechar()

`procedure, private mo_netcdf::ncdataset::setglobalattributechar ( )  [private]`

17.69.2.26 setglobalattributef32()

`procedure, private mo_netcdf::ncdataset::setglobalattributef32 ( )  [private]`

17.69.2.27 setglobalattributef64()

```
procedure, private mo_netcdf::ncdataset::setglobalattributef64 ( )  [private]
```

17.69.2.28 setglobalattributei16()

```
procedure, private mo_netcdf::ncdataset::setglobalattributei16 ( )  [private]
```

17.69.2.29 setglobalattributei32()

```
procedure, private mo_netcdf::ncdataset::setglobalattributei32 ( )  [private]
```

17.69.2.30 setglobalattributei64()

```
procedure, private mo_netcdf::ncdataset::setglobalattributei64 ( )  [private]
```

17.69.2.31 setglobalattributei8()

```
procedure, private mo_netcdf::ncdataset::setglobalattributei8 ( )  [private]
```

17.69.2.32 setvariable()

```
generic, public mo_netcdf::ncdataset::setvariable ( )
```

Create a NetCDF variable.

Create a NetCDF Variable with given name, data type and dimensions. All optional arguments to the nf90_def_var function are supported. The program will teminate abruptly if the variable cannot be created. Supported data types and their string encodings: NF90_BYTE -> "i8" NF90_SHORT -> "i16" NF90_INT -> "i32" NF90_INT64 -> "i64" NF90_FLOAT -> "f32" NF90_DOUBLE -> "f64"

Parameters

in	*character(∗) :: name*	
in	*character(3) :: dtype*	
in	*integer(i4)/character(∗)/type(NcDataset) :: dimensions(:)*	
in	*logical :: contiguous*	
in	*integer(i4) :: chunksizes(:)*	
in	*integer(i4) :: deflate_level*	
in	*logical :: shuffle*	
in	*logical :: fletcher32*	
in	*integer(i4) :: endianess*	
in	*integer(i4) :: cache_size*	
in	*integer(i4) :: cache_nelems*	
in	*integer(i4) :: cache_preemption*	

Returns

NcVariable

Author

David Schaefer

Date

June 2015

17.69.2.33 setvariablewithids()

```
procedure, private mo_netcdf::ncdataset::setvariablewithids ( )  [private]
```

17.69.2.34 setvariablewithnames()

```
procedure, private mo_netcdf::ncdataset::setvariablewithnames ( )  [private]
```

17.69.2.35 setvariablewithtypes()

```
procedure, private mo_netcdf::ncdataset::setvariablewithtypes ( )  [private]
```

17.69.3 Member Data Documentation

17.69.3.1 dataset

```
character(256) mo_netcdf::ncdataset::dataset
```

17.69.3.2 file

```
character(1) mo_netcdf::ncdataset::file
```

17.69.3.3 filename

```
character(256) mo_netcdf::ncdataset::filename
```

17.69.3.4 fname

```
character(256) mo_netcdf::ncdataset::fname
```

17.69.3.5 id

```
integer(i4) mo_netcdf::ncdataset::id
```

17.69.3.6 mode

```
character(1) mo_netcdf::ncdataset::mode
```

17.69.3.7 netcdf

```
integer(i4) mo_netcdf::ncdataset::netcdf
```

17.69.3.8 of

```
character(256) mo_netcdf::ncdataset::of
```

17.69.3.9 open

```
character(1) mo_netcdf::ncdataset::open
```

17.69.3.10 opened

```
character(256) mo_netcdf::ncdataset::opened
```

17.69.3.11 the

```
character(256) mo_netcdf::ncdataset::the
```

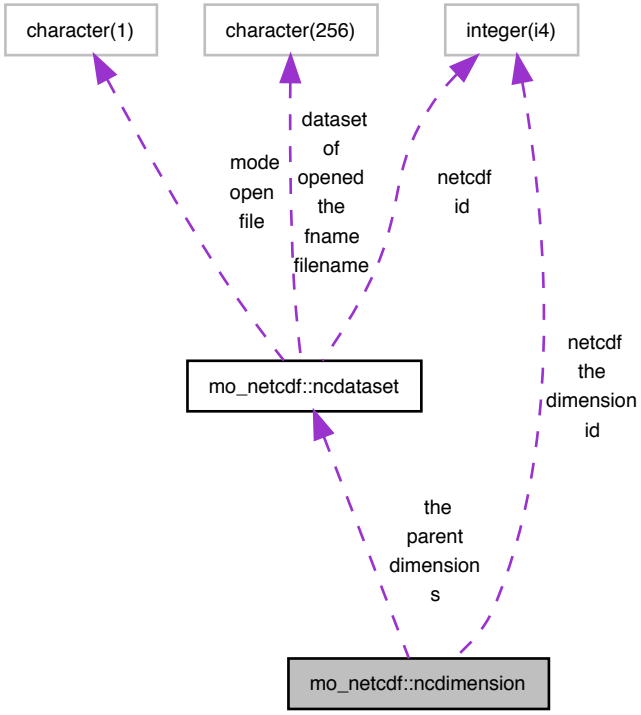
The documentation for this interface was generated from the following file:

- mo_netcdf.f90

17.70 mo_netcdf::ncdimension Type Reference

Provides the dimension access functionality.

Collaboration diagram for mo_netcdf::ncdimension:



Public Member Functions

- procedure, public initncvariable
- procedure, public getname => getVariableName
- procedure, public getnodimensions

  Retrieve the number of dimensions.
- procedure, public getdimensions => getVariableDimensions

  Retrieve the variable dimensions.
- procedure, public getshape => getVariableShape

  Retrieve the shape of the variable.
- procedure, public getdtype => getVariableDtype

  Retrieve the variable data type.
- procedure, public hasattribute

  Check if attribute exists.
- procedure, public isunlimited => isUnlimitedVariable

  Check if the variable is unlimited.
- generic, public setdata => setDataScalarI8, setData1dI8, setData2dI8, setData3dI8, setData4dI8, setData5dI8, setDataScalarI16, setData1dI16, setData2dI16, setData3dI16, setData4dI16, setData5dI16, setDataScalarI32, setData1dI32, setData2dI32, setData3dI32, setData4dI32, setData5dI32, setDataScalarI64, setData1dI64, setData2dI64, setData3dI64, setData4dI64, setData5dI64, setDataScalarF32, setData1dF32, setData2dF32, setData3dF32, setData4dF32, setData5dF32, setDataScalarF64, setData1dF64, setData2dF64, setData3dF64, setData4dF64, setData5dF64

  Write data to variable.

- generic, public getdata => getDataScalarI8, getData1dI8, getData2dI8, getData3dI8, getData4dI8, getData5dI8, getDataScalarI16, getData1dI16, getData2dI16, getData3dI16, getData4dI16, getData5dI16, getDataScalarI32, getData1dI32, getData2dI32, getData3dI32, getData4dI32, getData5dI32, getDataScalarI64, getData1dI64, getData2dI64, getData3dI64, getData4dI64, getData5dI64, getDataScalarF32, getData1dF32, getData2dF32, getData3dF32, getData4dF32, getData5dF32, getDataScalarF64, getData1dF64, getData2dF64, getData3dF64, getData4dF64, getData5dF64

  Retrieve data.
- generic, public setfillvalue => setVariableFillValueI8, setVariableFillValueI16, setVariableFillValueI32, setVariableFillValueI64, setVariableFillValueF32, setVariableFillValueF64

  Set the variable fill value.
- generic, public getfillvalue => getVariableFillValueI8, getVariableFillValueI16, getVariableFillValueI32, getVariableFillValueI64, getVariableFillValueF32, getVariableFillValueF64

  Retrieve the variable fill value.
- generic, public setattribute => setVariableAttributeChar, setVariableAttributeI8, setVariableAttributeI16, setVariableAttributeI32, setVariableAttributeI64, setVariableAttributeF32, setVariableAttributeF64

  Create a new variable attribute.
- generic, public getattribute => getVariableAttributeChar, getVariableAttributeI8, getVariableAttributeI16, getVariableAttributeI32, getVariableAttributeI64, getVariableAttributeF32, getVariableAttributeF64

  Retrieve variable attribute value.

Public Attributes

- integer(i4) id
- integer(i4) the
- integer(i4) netcdf
- integer(i4) dimension
- type(ncdataset) parent
- type(ncdataset) the
- type(ncdataset) dimension
- type(ncdataset) s

Private Member Functions

- procedure, private setvariableattributechar
- procedure, private setvariableattributei8
- procedure, private setvariableattributei16
- procedure, private setvariableattributei32
- procedure, private setvariableattributei64
- procedure, private setvariableattributef32
- procedure, private setvariableattributef64
- procedure, private getvariableattributechar
- procedure, private getvariableattributei8
- procedure, private getvariableattributei16
- procedure, private getvariableattributei32
- procedure, private getvariableattributei64
- procedure, private getvariableattributef32
- procedure, private getvariableattributef64
- procedure, private setdatascalari8
- procedure, private setdata1di8
- procedure, private setdata2di8
- procedure, private setdata3di8
- procedure, private setdata4di8
- procedure, private setdata5di8

- procedure, private setdatascalari16
- procedure, private setdata1di16
- procedure, private setdata2di16
- procedure, private setdata3di16
- procedure, private setdata4di16
- procedure, private setdata5di16
- procedure, private setdatascalari32
- procedure, private setdata1di32
- procedure, private setdata2di32
- procedure, private setdata3di32
- procedure, private setdata4di32
- procedure, private setdata5di32
- procedure, private setdatascalari64
- procedure, private setdata1di64
- procedure, private setdata2di64
- procedure, private setdata3di64
- procedure, private setdata4di64
- procedure, private setdata5di64
- procedure, private setdatascalarf32
- procedure, private setdata1df32
- procedure, private setdata2df32
- procedure, private setdata3df32
- procedure, private setdata4df32
- procedure, private setdata5df32
- procedure, private setdatascalarf64
- procedure, private setdata1df64
- procedure, private setdata2df64
- procedure, private setdata3df64
- procedure, private setdata4df64
- procedure, private setdata5df64
- procedure, private getdatascalari8
- procedure, private getdata1di8
- procedure, private getdata2di8
- procedure, private getdata3di8
- procedure, private getdata4di8
- procedure, private getdata5di8
- procedure, private getdatascalari16
- procedure, private getdata1di16
- procedure, private getdata2di16
- procedure, private getdata3di16
- procedure, private getdata4di16
- procedure, private getdata5di16
- procedure, private getdatascalari32
- procedure, private getdata1di32
- procedure, private getdata2di32
- procedure, private getdata3di32
- procedure, private getdata4di32
- procedure, private getdata5di32
- procedure, private getdatascalari64
- procedure, private getdata1di64
- procedure, private getdata2di64
- procedure, private getdata3di64
- procedure, private getdata4di64
- procedure, private getdata5di64
- procedure, private getdatascalarf32

- procedure, private getdata1df32
- procedure, private getdata2df32
- procedure, private getdata3df32
- procedure, private getdata4df32
- procedure, private getdata5df32
- procedure, private getdatascalarf64
- procedure, private getdata1df64
- procedure, private getdata2df64
- procedure, private getdata3df64
- procedure, private getdata4df64
- procedure, private getdata5df64
- procedure, private setvariablefillvaluei8
- procedure, private setvariablefillvaluei16
- procedure, private setvariablefillvaluei32
- procedure, private setvariablefillvaluei64
- procedure, private setvariablefillvaluef32
- procedure, private setvariablefillvaluef64
- procedure, private getvariablefillvaluei8
- procedure, private getvariablefillvaluei16
- procedure, private getvariablefillvaluei32
- procedure, private getvariablefillvaluei64
- procedure, private getvariablefillvaluef32
- procedure, private getvariablefillvaluef64

17.70.1 Detailed Description

Provides the dimension access functionality.

Bound to this derived type is some necessary inquire functionality. This type is not to be instantiated directly! Use the getDimension/setDimension functions of a NcDataset instance as a "constructor".

17.70.2 Member Function/Subroutine Documentation

17.70.2.1 getattribute()

`generic, public mo_netcdf::ncdimension::getattribute ( )`

Retrieve variable attribute value.

Retrieve the value for a variable attribute specified by its name. The program will teminate abruptly if the attribute does not exist.

Parameters

in	*character(∗) :: name*	
out	*character(∗)/integer(i4)/real(sp)/real(dp) :: value*	

Author

David Schaefer

Date

June 2015

17.70.2.2 getdata()

```
generic, public mo_netcdf::ncdimension::getdata ( )
```

Retrieve data.

Read the data from an optionally given position. All optional arguments to the nf90_get_var function are supported. A read error will result in abrupt program termination.

Parameters

in	*integer(i4) :: start(:), cnt(:), stride(:), map(:)*	
out	*integer(i4)/real(sp)/real(dp), allocatable, dimension(/(:)/(:,:)/(:,:,:)/(:,:,:,:)/(:,:,:,:,:)) :: values*	

Author

David Schaefer

Date

June 2015

17.70.2.3 getdata1df32()

```
procedure, private mo_netcdf::ncdimension::getdata1df32 ( )  [private]
```

17.70.2.4 getdata1df64()

```
procedure, private mo_netcdf::ncdimension::getdata1df64 ( )  [private]
```

17.70.2.5 getdata1di16()

```
procedure, private mo_netcdf::ncdimension::getdata1di16 ( )  [private]
```

17.70.2.6 getdata1di32()

```
procedure, private mo_netcdf::ncdimension::getdata1di32 ( )  [private]
```

17.70.2.7 getdata1di64()

```
procedure, private mo_netcdf::ncdimension::getdata1di64 ( )  [private]
```

17.70.2.8 getdata1di8()

```
procedure, private mo_netcdf::ncdimension::getdata1di8 ( )  [private]
```

17.70.2.9 getdata2df32()

```
procedure, private mo_netcdf::ncdimension::getdata2df32 ( )  [private]
```

17.70.2.10 getdata2df64()

```
procedure, private mo_netcdf::ncdimension::getdata2df64 ( )  [private]
```

17.70.2.11 getdata2di16()

```
procedure, private mo_netcdf::ncdimension::getdata2di16 ( )  [private]
```

17.70.2.12 getdata2di32()

```
procedure, private mo_netcdf::ncdimension::getdata2di32 ( )  [private]
```

17.70.2.13 getdata2di64()

```
procedure, private mo_netcdf::ncdimension::getdata2di64 ( )  [private]
```

17.70.2.14 getdata2di8()

```
procedure, private mo_netcdf::ncdimension::getdata2di8 ( )  [private]
```

17.70.2.15 getdata3df32()

```
procedure, private mo_netcdf::ncdimension::getdata3df32 ( )  [private]
```

17.70.2.16 getdata3df64()

```
procedure, private mo_netcdf::ncdimension::getdata3df64 ( )  [private]
```

17.70.2.17 getdata3di16()

```
procedure, private mo_netcdf::ncdimension::getdata3di16 ( )  [private]
```

17.70.2.18 getdata3di32()

```
procedure, private mo_netcdf::ncdimension::getdata3di32 ( )  [private]
```

17.70.2.19 getdata3di64()

```
procedure, private mo_netcdf::ncdimension::getdata3di64 ( )  [private]
```

17.70.2.20 getdata3di8()

```
procedure, private mo_netcdf::ncdimension::getdata3di8 ( )  [private]
```

17.70.2.21 getdata4df32()

```
procedure, private mo_netcdf::ncdimension::getdata4df32 ( )  [private]
```

17.70.2.22 getdata4df64()

```
procedure, private mo_netcdf::ncdimension::getdata4df64 ( )  [private]
```

17.70.2.23 getdata4di16()

```
procedure, private mo_netcdf::ncdimension::getdata4di16 ( )  [private]
```

17.70.2.24 getdata4di32()

```
procedure, private mo_netcdf::ncdimension::getdata4di32 ( )  [private]
```

17.70.2.25 getdata4di64()

```
procedure, private mo_netcdf::ncdimension::getdata4di64 ( )  [private]
```

17.70.2.26 getdata4di8()

```
procedure, private mo_netcdf::ncdimension::getdata4di8 ( )  [private]
```

17.70.2.27 getdata5df32()

```
procedure, private mo_netcdf::ncdimension::getdata5df32 ( )  [private]
```

17.70.2.28 getdata5df64()

```
procedure, private mo_netcdf::ncdimension::getdata5df64 ( )  [private]
```

17.70.2.29 getdata5di16()

```
procedure, private mo_netcdf::ncdimension::getdata5di16 ( )  [private]
```

17.70.2.30 getdata5di32()

```
procedure, private mo_netcdf::ncdimension::getdata5di32 ( )  [private]
```

17.70.2.31 getdata5di64()

```
procedure, private mo_netcdf::ncdimension::getdata5di64 ( )  [private]
```

17.70.2.32 getdata5di8()

```
procedure, private mo_netcdf::ncdimension::getdata5di8 ( )  [private]
```

17.70.2.33 getdatascalarf32()

```
procedure, private mo_netcdf::ncdimension::getdatascalarf32 ( )  [private]
```

17.70.2.34 getdatascalarf64()

```
procedure, private mo_netcdf::ncdimension::getdatascalarf64 ( )  [private]
```

17.70.2.35 getdatascalari16()

```
procedure, private mo_netcdf::ncdimension::getdatascalari16 ( )  [private]
```

17.70.2.36 getdatascalari32()

```
procedure, private mo_netcdf::ncdimension::getdatascalari32 ( )  [private]
```

17.70.2.37 getdatascalari64()

```
procedure, private mo_netcdf::ncdimension::getdatascalari64 ( )  [private]
```

17.70.2.38 getdatascalari8()

```
procedure, private mo_netcdf::ncdimension::getdatascalari8 ( )  [private]
```

17.70.2.39 getdimensions()

```
procedure, public mo_netcdf::ncdimension::getdimensions ( )
```

Retrieve the variable dimensions.

Return the ids of the dimensions associated with variable.

17.70.2.40 getdtype()

```
procedure, public mo_netcdf::ncdimension::getdtype ( )
```

Retrieve the variable data type.

Return the encoded data type of the variable. Data type encodeings "f32" -> NF90_FLOAT "f64" -> NF90_DOUBLE "i8" -> NF90_BYTE "i16" -> NF90_SHORT "i32" -> NF90_INT "i64" -> NF90_INT64

17.70.2.41 getfillvalue()

```
generic, public mo_netcdf::ncdimension::getfillvalue ( )
```

Retrieve the variable fill value.

Retrieve the variable fill value or a default value if fill value was not explicitly set. A read error results in abrupt program temination.

Parameters

out	*integer(i4)/real(sp)/real(dp) :: fvalue*	

Author

David Schaefer

Date

June 2015

17.70.2.42 getname()

```
procedure, public mo_netcdf::ncdimension::getname ( )
```

17.70.2.43 getnodimensions()

```
procedure, public mo_netcdf::ncdimension::getnodimensions ( )
```

Retrieve the number of dimensions.

Return the number of dimensions associated with variable

17.70.2.44 getshape()

```
procedure, public mo_netcdf::ncdimension::getshape ( )
```

Retrieve the shape of the variable.

Return the shape of the variable.

17.70.2.45 getvariableattributechar()

```
procedure, private mo_netcdf::ncdimension::getvariableattributechar ( )  [private]
```

17.70.2.46 getvariableattributef32()

```
procedure, private mo_netcdf::ncdimension::getvariableattributef32 ( )  [private]
```

17.70.2.47 getvariableattributef64()

```
procedure, private mo_netcdf::ncdimension::getvariableattributef64 ( )  [private]
```

17.70.2.48 getvariableattributei16()

```
procedure, private mo_netcdf::ncdimension::getvariableattributei16 ( )  [private]
```

17.70.2.49 getvariableattributei32()

```
procedure, private mo_netcdf::ncdimension::getvariableattributei32 ( )  [private]
```

17.70.2.50 getvariableattributei64()

```
procedure, private mo_netcdf::ncdimension::getvariableattributei64 ( )  [private]
```

17.70.2.51 getvariableattributei8()

```
procedure, private mo_netcdf::ncdimension::getvariableattributei8 ( )  [private]
```

17.70.2.52 getvariablefillvaluef32()

```
procedure, private mo_netcdf::ncdimension::getvariablefillvaluef32 ( )  [private]
```

17.70.2.53 getvariablefillvaluef64()

```
procedure, private mo_netcdf::ncdimension::getvariablefillvaluef64 ( )  [private]
```

17.70.2.54 getvariablefillvaluei16()

```
procedure, private mo_netcdf::ncdimension::getvariablefillvaluei16 ( )  [private]
```

17.70.2.55 getvariablefillvaluei32()

```
procedure, private mo_netcdf::ncdimension::getvariablefillvaluei32 ( )  [private]
```

17.70.2.56 getvariablefillvaluei64()

```
procedure, private mo_netcdf::ncdimension::getvariablefillvaluei64 ( )  [private]
```

17.70.2.57 getvariablefillvaluei8()

```
procedure, private mo_netcdf::ncdimension::getvariablefillvaluei8 ( )  [private]
```

17.70.2.58 hasattribute()

```
procedure, public mo_netcdf::ncdimension::hasattribute ( )
```

Check if attribute exists.

Returns true if an attribute with the given name exists, false otherwise.

17.70.2.59 initncvariable()

```
procedure, public mo_netcdf::ncdimension::initncvariable ( )
```

17.70.2.60 isunlimited()

```
procedure, public mo_netcdf::ncdimension::isunlimited ( )
```

Check if the variable is unlimited.

Returns true if the variable has an unlimited dimension, false otherwise.

Returns

"logical"

Author

David Schaefer

Date

June 2015

17.70.2.61 setattribute()

```
generic, public mo_netcdf::ncdimension::setattribute ( )
```

Create a new variable attribute.

Create a new variable attribute from given name and value. A write error results in abrupt program temination.

Parameters

in	*character(∗) :: name*	
in	*character(∗)/integer(i4)/real(sp)/real(dp) :: value*	

Author

David Schaefer

Date

June 2015

17.70.2.62 setdata()

```
generic, public mo_netcdf::ncdimension::setdata ( )
```

Write data to variable.

Write the given data into the variable at an optionally given position. All optional arguments to the nf90_put_var function are supported. A write error will result in abrupt program termination.

Parameters

in	*integer(i4)/real(sp)/real(dp), dimension(/(:)/(:,:)/(:,:,:)/(:,:,:,:)/(:,:,:,:,:)) :: values*	
in	*integer(i4) :: start(:), cnt(:), stride(:), map(:)*	

Author

David Schaefer

Date

June 2015

17.70.2.63 setdata1df32()

```
procedure, private mo_netcdf::ncdimension::setdata1df32 ( )  [private]
```

17.70.2.64 setdata1df64()

```
procedure, private mo_netcdf::ncdimension::setdata1df64 ( )  [private]
```

17.70.2.65 setdata1di16()

```
procedure, private mo_netcdf::ncdimension::setdata1di16 ( )  [private]
```

17.70.2.66 setdata1di32()

```
procedure, private mo_netcdf::ncdimension::setdata1di32 ( )  [private]
```

17.70.2.67 setdata1di64()

```
procedure, private mo_netcdf::ncdimension::setdata1di64 ( )  [private]
```

17.70.2.68 setdata1di8()

```
procedure, private mo_netcdf::ncdimension::setdata1di8 ( )  [private]
```

17.70.2.69 setdata2df32()

```
procedure, private mo_netcdf::ncdimension::setdata2df32 ( )  [private]
```

17.70.2.70 setdata2df64()

```
procedure, private mo_netcdf::ncdimension::setdata2df64 ( )  [private]
```

17.70.2.71 setdata2di16()

```
procedure, private mo_netcdf::ncdimension::setdata2di16 ( )  [private]
```

17.70.2.72 setdata2di32()

```
procedure, private mo_netcdf::ncdimension::setdata2di32 ( )  [private]
```

17.70.2.73 setdata2di64()

```
procedure, private mo_netcdf::ncdimension::setdata2di64 ( )  [private]
```

17.70.2.74 setdata2di8()

```
procedure, private mo_netcdf::ncdimension::setdata2di8 ( )  [private]
```

17.70.2.75 setdata3df32()

```
procedure, private mo_netcdf::ncdimension::setdata3df32 ( )  [private]
```

17.70.2.76 setdata3df64()

```
procedure, private mo_netcdf::ncdimension::setdata3df64 ( )  [private]
```

17.70.2.77 setdata3di16()

```
procedure, private mo_netcdf::ncdimension::setdata3di16 ( )  [private]
```

17.70.2.78 setdata3di32()

```
procedure, private mo_netcdf::ncdimension::setdata3di32 ( )  [private]
```

17.70.2.79 setdata3di64()

```
procedure, private mo_netcdf::ncdimension::setdata3di64 ( )  [private]
```

17.70.2.80 setdata3di8()

```
procedure, private mo_netcdf::ncdimension::setdata3di8 ( )  [private]
```

17.70.2.81 setdata4df32()

```
procedure, private mo_netcdf::ncdimension::setdata4df32 ( )  [private]
```

17.70.2.82 setdata4df64()

```
procedure, private mo_netcdf::ncdimension::setdata4df64 ( )  [private]
```

17.70.2.83 setdata4di16()

```
procedure, private mo_netcdf::ncdimension::setdata4di16 ( )  [private]
```

17.70.2.84 setdata4di32()

```
procedure, private mo_netcdf::ncdimension::setdata4di32 ( )  [private]
```

17.70.2.85 setdata4di64()

```
procedure, private mo_netcdf::ncdimension::setdata4di64 ( )  [private]
```

17.70.2.86 setdata4di8()

```
procedure, private mo_netcdf::ncdimension::setdata4di8 ( )  [private]
```

17.70.2.87 setdata5df32()

```
procedure, private mo_netcdf::ncdimension::setdata5df32 ( )  [private]
```

17.70.2.88 setdata5df64()

```
procedure, private mo_netcdf::ncdimension::setdata5df64 ( )  [private]
```

17.70.2.89 setdata5di16()

```
procedure, private mo_netcdf::ncdimension::setdata5di16 ( )  [private]
```

17.70.2.90 setdata5di32()

```
procedure, private mo_netcdf::ncdimension::setdata5di32 ( )  [private]
```

17.70.2.91 setdata5di64()

```
procedure, private mo_netcdf::ncdimension::setdata5di64 ( )  [private]
```

17.70.2.92 setdata5di8()

```
procedure, private mo_netcdf::ncdimension::setdata5di8 ( )  [private]
```

17.70.2.93 setdatascalarf32()

```
procedure, private mo_netcdf::ncdimension::setdatascalarf32 ( )  [private]
```

17.70.2.94 setdatascalarf64()

```
procedure, private mo_netcdf::ncdimension::setdatascalarf64 ( )  [private]
```

17.70.2.95 setdatascalari16()

```
procedure, private mo_netcdf::ncdimension::setdatascalari16 ( )  [private]
```

17.70.2.96 setdatascalari32()

```
procedure, private mo_netcdf::ncdimension::setdatascalari32 ( )  [private]
```

17.70.2.97 setdatascalari64()

```
procedure, private mo_netcdf::ncdimension::setdatascalari64 ( )  [private]
```

17.70.2.98 setdatascalari8()

```
procedure, private mo_netcdf::ncdimension::setdatascalari8 ( )  [private]
```

17.70.2.99 setfillvalue()

```
generic, public mo_netcdf::ncdimension::setfillvalue ( )
```

Set the variable fill value.

Define the variable fill value. A write error results in abrupt program temination.

Note

This procedure must be called AFTER the variable was created but BEFORE data is first written.

Parameters

in	*integer(i4)/real(sp)/real(dp) :: fvalue*	

Author

David Schaefer

Date

June 2015

17.70.2.100 setvariableattributechar()

```
procedure, private mo_netcdf::ncdimension::setvariableattributechar ( )  [private]
```

17.70.2.101 setvariableattributef32()

```
procedure, private mo_netcdf::ncdimension::setvariableattributef32 ( )  [private]
```

17.70.2.102 setvariableattributef64()

```
procedure, private mo_netcdf::ncdimension::setvariableattributef64 ( )  [private]
```

17.70.2.103 setvariableattributei16()

```
procedure, private mo_netcdf::ncdimension::setvariableattributei16 ( )  [private]
```

17.70.2.104 setvariableattributei32()

```
procedure, private mo_netcdf::ncdimension::setvariableattributei32 ( )  [private]
```

17.70.2.105 setvariableattributei64()

```
procedure, private mo_netcdf::ncdimension::setvariableattributei64 ( )  [private]
```

17.70.2.106 setvariableattributei8()

```
procedure, private mo_netcdf::ncdimension::setvariableattributei8 ( )  [private]
```

17.70.2.107 setvariablefillvaluef32()

```
procedure, private mo_netcdf::ncdimension::setvariablefillvaluef32 ( )  [private]
```

17.70.2.108 setvariablefillvaluef64()

```
procedure, private mo_netcdf::ncdimension::setvariablefillvaluef64 ( )  [private]
```

17.70.2.109 setvariablefillvaluei16()

```
procedure, private mo_netcdf::ncdimension::setvariablefillvaluei16 ( )  [private]
```

17.70.2.110 setvariablefillvaluei32()

```
procedure, private mo_netcdf::ncdimension::setvariablefillvaluei32 ( )  [private]
```

17.70.2.111 setvariablefillvaluei64()

```
procedure, private mo_netcdf::ncdimension::setvariablefillvaluei64 ( )  [private]
```

17.70.2.112 setvariablefillvaluei8()

```
procedure, private mo_netcdf::ncdimension::setvariablefillvaluei8 ( )  [private]
```

17.70.3 Member Data Documentation

17.70.3.1 dimension [1/2]

```
integer(i4) mo_netcdf::ncdimension::dimension
```

17.70.3.2 dimension [2/2]

```
type(ncdataset) mo_netcdf::ncdimension::dimension
```

17.70.3.3 id

```
integer(i4) mo_netcdf::ncdimension::id
```

17.70.3.4 netcdf

```
integer(i4) mo_netcdf::ncdimension::netcdf
```

17.70.3.5 parent

```
type(ncdataset) mo_netcdf::ncdimension::parent
```

17.70.3.6 s

```
type(ncdataset) mo_netcdf::ncdimension::s
```

17.70.3.7 the [1/2]

```
integer(i4) mo_netcdf::ncdimension::the
```

17.70.3.8 the [2/2]

```
type(ncdataset) mo_netcdf::ncdimension::the
```

The documentation for this type was generated from the following file:

- mo_netcdf.f90

17.71 mo_netcdf::ncvariable Interface Reference

The documentation for this interface was generated from the following file:

- mo_netcdf.f90

17.72 mo_utils::ne Interface Reference

Public Member Functions

- elemental pure logical function notequal_sp (a, b)
- elemental pure logical function notequal_dp (a, b)

17.72.1 Member Function/Subroutine Documentation

17.72.1.1 notequal_dp()

```
elemental pure logical function mo_utils::ne::notequal_dp (
             real(dp), intent(in) a,
             real(dp), intent(in) b )
```

17.72.1.2 notequal_sp()

```
elemental pure logical function mo_utils::ne::notequal_sp (
             real(sp), intent(in) a,
             real(sp), intent(in) b )
```

The documentation for this interface was generated from the following file:

- mo_utils.f90

17.73 mo_orderpack::nearless Interface Reference

Public Member Functions

- real(kind=dp) function d_nearless (XVAL)
- real(kind=sp) function r_nearless (XVAL)
- integer(kind=i4) function i_nearless (XVAL)

17.73.1 Member Function/Subroutine Documentation

17.73.1.1 d_nearless()

```
real(kind = dp) function mo_orderpack::nearless::d_nearless (
             real(kind = dp), intent(in) XVAL )
```

17.73.1.2 i_nearless()

```
integer(kind = i4) function mo_orderpack::nearless::i_nearless (
             integer(kind = i4), intent(in) XVAL )
```

17.73.1.3 r_nearless()

```
real(kind = sp) function mo_orderpack::nearless::r_nearless (
             real(kind = sp), intent(in) XVAL )
```

The documentation for this interface was generated from the following file:

- mo_orderpack.f90

17.74 mo_spatialsimilarity::nndv Interface Reference

Calculates the number of neighboring dominating values, a measure for spatial dissimilarity.

Public Member Functions

- real(sp) function nndv_sp (mat1, mat2, mask, valid)
- real(dp) function nndv_dp (mat1, mat2, mask, valid)

17.74.1 Detailed Description

Calculates the number of neighboring dominating values, a measure for spatial dissimilarity.

NNDV = 1 - sum(abs(dominating_neighbors(mat1) - dominating_neighbors(mat2))) / count(mask) dominating_↩ neighbors(mat1) = comparison if pixel is larger than its neighbouring values

An array element value is compared with its 8 neighbouring cells to check if these cells are larger than the array element value. The result is a 3x3 matrix in which larger cells are indicated by a true value. For comparison this is done with both input arrays. The resulting array is the sum of substraction of the 3x3 matrices for each of the both

arrays. The resulting matrix is afterwards normalized to its available neighbors. Furthermore an average over the entire field is calculated. The valid interval of the values for NNDV is [0..1]. In which 1 indicates full agreement and 0 full dismatching.

```
EXAMPLE:~

mat1 =  | 12 17  1 | , mat2 = |  7  9 12 |
        |  4 10 11 |          | 12 11 11 |
        | 15  2 20 |          |  5 13  7 |
booleans determined for every grid cell following fortran array scrolling
i.e. (/col1_row1, col1_row2, col1_row3, col2_row1, .. ,col3_row3/),(/3,3/)

comp1 = | FFF FFF FTF, FFF FFF FFF, FTT FFT FFF |
        | FFF TFT TTF, TFT TFF FTT, TFF FFT FFF |
        | FFF FFF FFF, TTF TFF TTF, FFF FFF FFF |

comp2 = | FFF FFT FTT, FFT FFT FTT, FFF FFF FFF |
        | FFF FFF FFT, FTF FFT TFF, FFT TFF FFF |
        | FFF TFF TTF, FFF FFF FFF, TTF TFF FFF |

NNDVMatrix =
abs( count(comp1) - count(comp2) ) = | 1-3, 0-4, 3-0 | = | 2, 4, 3 |
                                     | 4-1, 5-3, 2-2 |   | 3, 2, 0 |
                                     | 0-3, 5-0, 0-3 |   | 3, 5, 3 |

                            DISSIMILAR / VALID NEIGH CELLS
NNDVMatrix / VALID NEIGH CELLS = | 2, 4, 3 | / | 3, 5, 3 |
                                 | 3, 2, 0 |   | 5, 8, 5 |
                                 | 3, 5, 3 |   | 3, 5, 3 |

                               = | 0.66, 0.80, 1.00 |
                                 | 0.60, 0.25, 0.00 |
                                 | 1.0,  1.00, 1.00 |

NNDV = 1 - sum(NNDVMatrix) / count(mask) = 1 - (6.31666666 / 9) = 0.2981
```

If an optinal mask is given, the calculations are over those locations that correspond to true values in the mask. mat1 and mat2 can be single or double precision. The result will have the same numerical precision.

Parameters

in	*real(sp/dp), dimension(:,:) :: mat1*	2D-array with input numbers
in	*real(sp/dp), dimension(:,:) :: mat2*	2D-array with input numbers
in	*logical,dimension(:,:),optinal :: mask*	2D-array of logical values with size(mat1/mat2). If present, only those locations in mat1/mat2 having true values in mask are evaluated.
out	*logical :: valid*	indicates if the function could determine a valid value result can be unvalid if entire mask is .false. for ex. in this case PatternDissim is set to 0 (worst case)

Returns

real(sp/dp) :: NNDV — Number of neighboring dominating values

Note

routine based on algorithm by Luis Samaniego 2009

Author

Matthias Zink

Date

Nov 2012

17.74.2 Member Function/Subroutine Documentation

17.74.2.1 nndv_dp()

```
real(dp) function mo_spatialsimilarity::nndv::nndv_dp (
             real(dp), dimension(:, :), intent(in) mat1,
             real(dp), dimension(:, :), intent(in) mat2,
             logical, dimension(:, :), intent(in), optional mask,
             logical, intent(out), optional valid )
```

17.74.2.2 nndv_sp()

```
real(sp) function mo_spatialsimilarity::nndv::nndv_sp (
             real(sp), dimension(:, :), intent(in) mat1,
             real(sp), dimension(:, :), intent(in) mat2,
             logical, dimension(:, :), intent(in), optional mask,
             logical, intent(out), optional valid )
```

The documentation for this interface was generated from the following file:

- mo_spatialsimilarity.f90

17.75 mo_utils::notequal Interface Reference

Public Member Functions

- elemental pure logical function notequal_sp (a, b)
- elemental pure logical function notequal_dp (a, b)

17.75.1 Member Function/Subroutine Documentation

17.75.1.1 notequal_dp()

```
elemental pure logical function mo_utils::notequal::notequal_dp (
             real(dp), intent(in) a,
             real(dp), intent(in) b )
```

17.75.1.2 notequal_sp()

```
elemental pure logical function mo_utils::notequal::notequal_sp (
            real(sp), intent(in) a,
            real(sp), intent(in) b )
```

The documentation for this interface was generated from the following file:

- mo_utils.f90

17.76 mo_errormeasures::nse Interface Reference

Public Member Functions

- real(sp) function nse_sp_1d (x, y, mask)
- real(dp) function nse_dp_1d (x, y, mask)
- real(dp) function nse_dp_2d (x, y, mask)
- real(sp) function nse_sp_2d (x, y, mask)
- real(sp) function nse_sp_3d (x, y, mask)
- real(dp) function nse_dp_3d (x, y, mask)

17.76.1 Member Function/Subroutine Documentation

17.76.1.1 nse_dp_1d()

```
real(dp) function mo_errormeasures::nse::nse_dp_1d (
            real(dp), dimension(:), intent(in) x,
            real(dp), dimension(:), intent(in) y,
            logical, dimension(:), intent(in), optional mask )
```

17.76.1.2 nse_dp_2d()

```
real(dp) function mo_errormeasures::nse::nse_dp_2d (
            real(dp), dimension(:, :), intent(in) x,
            real(dp), dimension(:, :), intent(in) y,
            logical, dimension(:, :), intent(in), optional mask )
```

17.76.1.3 nse_dp_3d()

```
real(dp) function mo_errormeasures::nse::nse_dp_3d (
            real(dp), dimension(:, :, :), intent(in) x,
            real(dp), dimension(:, :, :), intent(in) y,
            logical, dimension(:, :, :), intent(in), optional mask )
```

17.76.1.4 nse_sp_1d()

```
real(sp) function mo_errormeasures::nse::nse_sp_1d (
             real(sp), dimension(:), intent(in) x,
             real(sp), dimension(:), intent(in) y,
             logical, dimension(:), intent(in), optional mask )
```

17.76.1.5 nse_sp_2d()

```
real(sp) function mo_errormeasures::nse::nse_sp_2d (
             real(sp), dimension(:, :), intent(in) x,
             real(sp), dimension(:, :), intent(in) y,
             logical, dimension(:, :), intent(in), optional mask )
```

17.76.1.6 nse_sp_3d()

```
real(sp) function mo_errormeasures::nse::nse_sp_3d (
             real(sp), dimension(:, :, :), intent(in) x,
             real(sp), dimension(:, :, :), intent(in) y,
             logical, dimension(:, :, :), intent(in), optional mask )
```

The documentation for this interface was generated from the following file:

- mo_errormeasures.f90

17.77 mo_string_utils::num2str Interface Reference

Convert to string.

Public Member Functions

- pure character(len=10) function i42str (nn, form)
- pure character(len=20) function i82str (nn, form)
- pure character(len=32) function sp2str (rr, form)
- pure character(len=32) function dp2str (rr, form)
- pure character(len=10) function log2str (ll, form)

17.77.1 Detailed Description

Convert to string.

Convert a number or logical to a string with an optional format.

Parameters

in	*integer(i4/i8)/real(sp/dp)/logical :: num*	Number or logical
in	*character(len=∗), optinal :: form*	Format string Defaults are: i4 - '(I10)' i8 - '(I20)' sp/dp - '(G32.5)' log - '(L10)'

Returns

character(len=X) :: str — String of formatted input number or logical
Ouput length X is:
i4 - 10
i8 - 20
sp/dp - 32
log - 10

Author

Matthias Cuntz - modified from Echam5, (C) MPI-MET, Hamburg, Germany

Date

Dec 2011

17.77.2 Member Function/Subroutine Documentation

17.77.2.1 dp2str()

```
pure character(len = 32) function mo_string_utils::num2str::dp2str (
             real(dp), intent(in) rr,
             character(len = *), intent(in), optional form )
```

17.77.2.2 i42str()

```
pure character(len = 10) function mo_string_utils::num2str::i42str (
             integer(i4), intent(in) nn,
             character(len = *), intent(in), optional form )
```

17.77.2.3 i82str()

```
pure character(len = 20) function mo_string_utils::num2str::i82str (
             integer(i8), intent(in) nn,
             character(len = *), intent(in), optional form )
```

17.77.2.4 log2str()

```
pure character(len = 10) function mo_string_utils::num2str::log2str (
             logical, intent(in) ll,
             character(len = *), intent(in), optional form )
```

17.77.2.5 sp2str()

```
pure character(len = 32) function mo_string_utils::num2str::sp2str (
            real(sp), intent(in) rr,
            character(len = *), intent(in), optional form )
```

The documentation for this interface was generated from the following file:

- mo_string_utils.f90

17.78 mo_string_utils::numarray2str Interface Reference

Convert to string.

Public Member Functions

- character(len=size(arr)) function i4array2str (arr)

17.78.1 Detailed Description

Convert to string.

Convert a array of numbers or logicals to a string.

Parameters

in	*integer(i4/i8)/real(sp/dp)/logical :: num(:)*	Array of numbers or logicals

Returns

character(len=X) :: str — String of formatted input number or logical

Author

Matthias Cuntz - modified from Echam5, (C) MPI-MET, Hamburg, Germany

Date

Dec 2011

17.78.2 Member Function/Subroutine Documentation

17.78.2.1 i4array2str()

```
character(len = size(arr)) function mo_string_utils::numarray2str::i4array2str (
            integer(i4), dimension(:), intent(in) arr )
```

The documentation for this interface was generated from the following file:

- mo_string_utils.f90

17.79 mo_optimization_utils::objective_interface Interface Reference

Public Member Functions

- real(dp) function objective_interface (parameterset, eval, arg1, arg2, arg3)

17.79.1 Constructor & Destructor Documentation

17.79.1.1 objective_interface()

```
real(dp) function mo_optimization_utils::objective_interface::objective_interface (
             real(dp), dimension(:), intent(in) parameterset,
             procedure(eval_interface), intent(in), pointer eval,
             real(dp), intent(in), optional arg1,
             real(dp), intent(out), optional arg2,
             real(dp), intent(out), optional arg3 )
```

References mo_kind::dp.

The documentation for this interface was generated from the following file:

- mo_optimization_utils.f90

17.80 mo_orderpack::omedian Interface Reference

Public Member Functions

- real(kind=dp) function d_median (XDONT)
- real(kind=sp) function r_median (XDONT)
- integer(kind=i4) function i_median (XDONT)

17.80.1 Member Function/Subroutine Documentation

17.80.1.1 d_median()

```
real(kind = dp) function mo_orderpack::omedian::d_median (
             real(kind = dp), dimension (:), intent(in) XDONT )
```

17.80.1.2 i_median()

```
integer(kind = i4) function mo_orderpack::omedian::i_median (
             integer(kind = i4), dimension (:), intent(in) XDONT )
```

17.80.1.3 r_median()

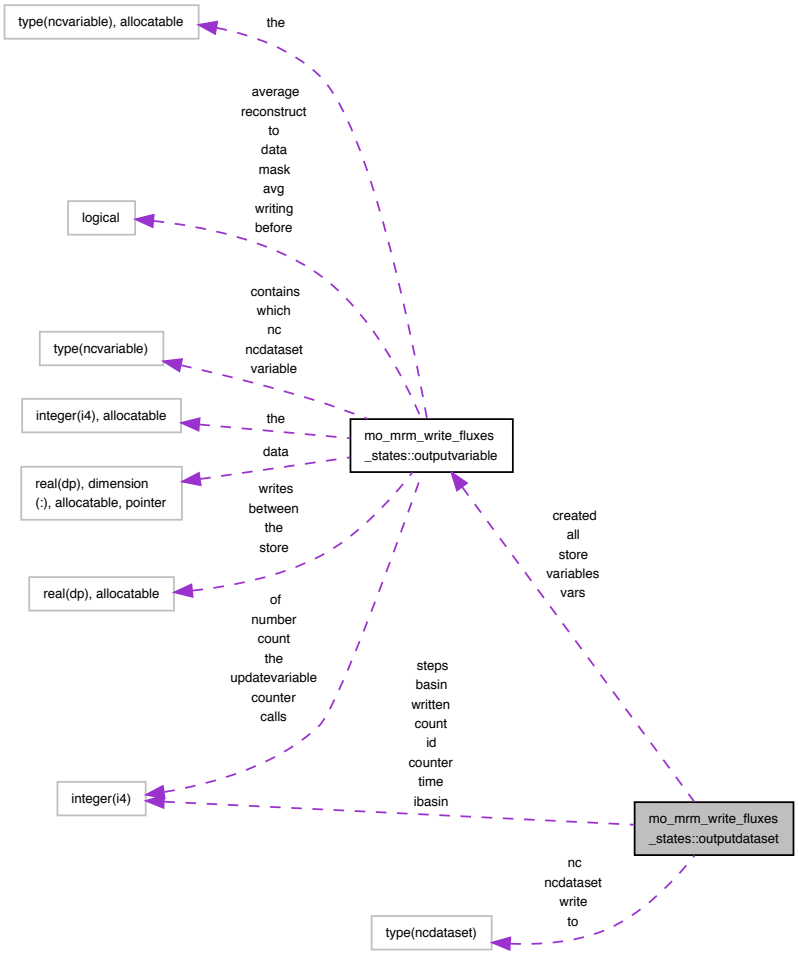
```
real(kind = sp) function mo_orderpack::omedian::r_median (
            real(kind = sp), dimension (:), intent(in) XDONT )
```

The documentation for this interface was generated from the following file:

- mo_orderpack.f90

17.81 mo_mrm_write_fluxes_states::outputdataset Interface Reference

Collaboration diagram for mo_mrm_write_fluxes_states::outputdataset:



Public Member Functions

- procedure, public updatedataset
- procedure, public writetimestep
- procedure, public close

Public Attributes

- integer(i4) ibasin
- integer(i4) basin
- integer(i4) id
- type(ncdataset) nc
- type(ncdataset) ncdataset
- type(ncdataset) to
- type(ncdataset) write
- type(outputvariable), dimension(:), allocatable vars
- type(outputvariable), allocatable store
- type(outputvariable), allocatable all
- type(outputvariable), dimension(dynamic), allocatable created
- type(outputvariable), allocatable variables
- integer(i4) counter = 0
- integer(i4) count
- integer(i4) written
- integer(i4) time
- integer(i4) steps

17.81.1 Member Function/Subroutine Documentation

17.81.1.1 close()

```
procedure, public mo_mrm_write_fluxes_states::outputdataset::close ( )
```

17.81.1.2 updatedataset()

```
procedure, public mo_mrm_write_fluxes_states::outputdataset::updatedataset ( )
```

17.81.1.3 writetimestep()

```
procedure, public mo_mrm_write_fluxes_states::outputdataset::writetimestep ( )
```

17.81.2 Member Data Documentation

17.81.2.1 all

```
type(outputvariable), allocatable mo_mrm_write_fluxes_states::outputdataset::all
```

17.81.2.2 basin

```
integer(i4) mo_mrm_write_fluxes_states::outputdataset::basin
```

17.81.2.3 count

integer(i4) mo_mrm_write_fluxes_states::outputdataset::count

17.81.2.4 counter

integer(i4) mo_mrm_write_fluxes_states::outputdataset::counter = 0

17.81.2.5 created

type(outputvariable), dimension (dynamic), allocatable mo_mrm_write_fluxes_states::outputdataset↩
::created

17.81.2.6 ibasin

integer(i4) mo_mrm_write_fluxes_states::outputdataset::ibasin

17.81.2.7 id

integer(i4) mo_mrm_write_fluxes_states::outputdataset::id

17.81.2.8 nc

type(ncdataset) mo_mrm_write_fluxes_states::outputdataset::nc

17.81.2.9 ncdataset

type(ncdataset) mo_mrm_write_fluxes_states::outputdataset::ncdataset

17.81.2.10 steps

integer(i4) mo_mrm_write_fluxes_states::outputdataset::steps

17.81.2.11 store

type(outputvariable), allocatable mo_mrm_write_fluxes_states::outputdataset::store

17.81.2.12 time

```
integer(i4) mo_mrm_write_fluxes_states::outputdataset::time
```

17.81.2.13 to

```
type(ncdataset) mo_mrm_write_fluxes_states::outputdataset::to
```

17.81.2.14 variables

```
type(outputvariable), allocatable mo_mrm_write_fluxes_states::outputdataset::variables
```

17.81.2.15 vars

```
type(outputvariable), dimension(:), allocatable mo_mrm_write_fluxes_states::outputdataset↩
::vars
```

17.81.2.16 write

```
type(ncdataset) mo_mrm_write_fluxes_states::outputdataset::write
```

17.81.2.17 written

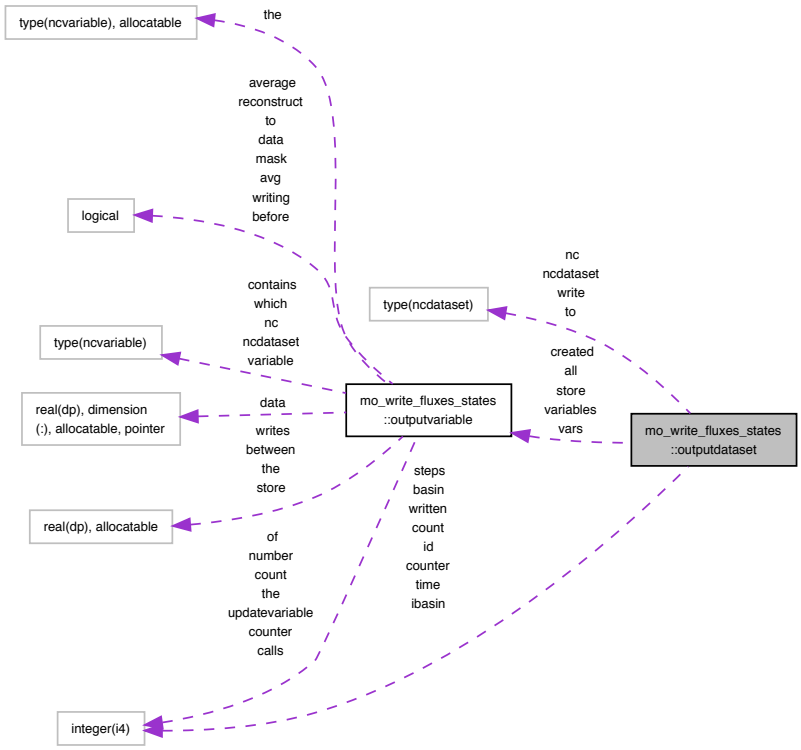
```
integer(i4) mo_mrm_write_fluxes_states::outputdataset::written
```

The documentation for this interface was generated from the following file:

- mo_mrm_write_fluxes_states.f90

17.82 mo_write_fluxes_states::outputdataset Interface Reference

Collaboration diagram for mo_write_fluxes_states::outputdataset:



Public Member Functions

- procedure, public updatedataset
- procedure, public writetimestep
- procedure, public close

Public Attributes

- integer(i4) ibasin
- integer(i4) basin
- integer(i4) id
- type(ncdataset) nc
- type(ncdataset) ncdataset
- type(ncdataset) to
- type(ncdataset) write
- type(outputvariable), dimension(:), allocatable vars
- type(outputvariable), allocatable store
- type(outputvariable), allocatable all
- type(outputvariable), dimension(dynamic), allocatable created
- type(outputvariable), allocatable variables
- integer(i4) counter = 0
- integer(i4) count

- integer(i4) written
- integer(i4) time
- integer(i4) steps

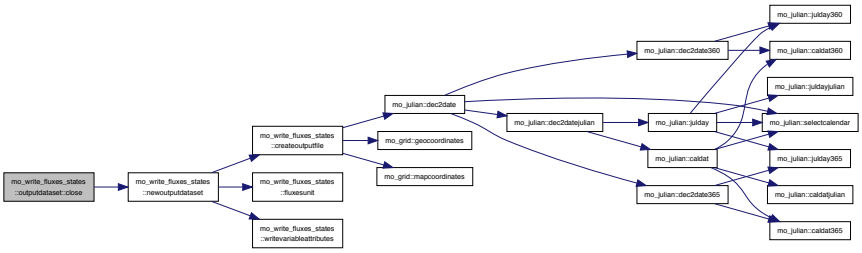
17.82.1 Member Function/Subroutine Documentation

17.82.1.1 close()

```
procedure, public mo_write_fluxes_states::outputdataset::close ( )
```

References mo_write_fluxes_states::newoutputdataset().

Here is the call graph for this function:



17.82.1.2 updatedataset()

```
procedure, public mo_write_fluxes_states::outputdataset::updatedataset ( )
```

17.82.1.3 writetimestep()

```
procedure, public mo_write_fluxes_states::outputdataset::writetimestep ( )
```

17.82.2 Member Data Documentation

17.82.2.1 all

```
type(outputvariable), allocatable mo_write_fluxes_states::outputdataset::all
```

17.82.2.2 basin

```
integer(i4) mo_write_fluxes_states::outputdataset::basin
```

17.82.2.3 count

integer(i4) mo_write_fluxes_states::outputdataset::count

17.82.2.4 counter

integer(i4) mo_write_fluxes_states::outputdataset::counter = 0

17.82.2.5 created

type(outputvariable), dimension (dynamic), allocatable mo_write_fluxes_states::outputdataset↩
::created

17.82.2.6 ibasin

integer(i4) mo_write_fluxes_states::outputdataset::ibasin

17.82.2.7 id

integer(i4) mo_write_fluxes_states::outputdataset::id

17.82.2.8 nc

type(ncdataset) mo_write_fluxes_states::outputdataset::nc

17.82.2.9 ncdataset

type(ncdataset) mo_write_fluxes_states::outputdataset::ncdataset

17.82.2.10 steps

integer(i4) mo_write_fluxes_states::outputdataset::steps

17.82.2.11 store

type(outputvariable), allocatable mo_write_fluxes_states::outputdataset::store

17.82.2.12 time

```
integer(i4) mo_write_fluxes_states::outputdataset::time
```

17.82.2.13 to

```
type(ncdataset) mo_write_fluxes_states::outputdataset::to
```

17.82.2.14 variables

```
type(outputvariable), allocatable mo_write_fluxes_states::outputdataset::variables
```

17.82.2.15 vars

```
type(outputvariable), dimension(:), allocatable mo_write_fluxes_states::outputdataset::vars
```

17.82.2.16 write

```
type(ncdataset) mo_write_fluxes_states::outputdataset::write
```

17.82.2.17 written

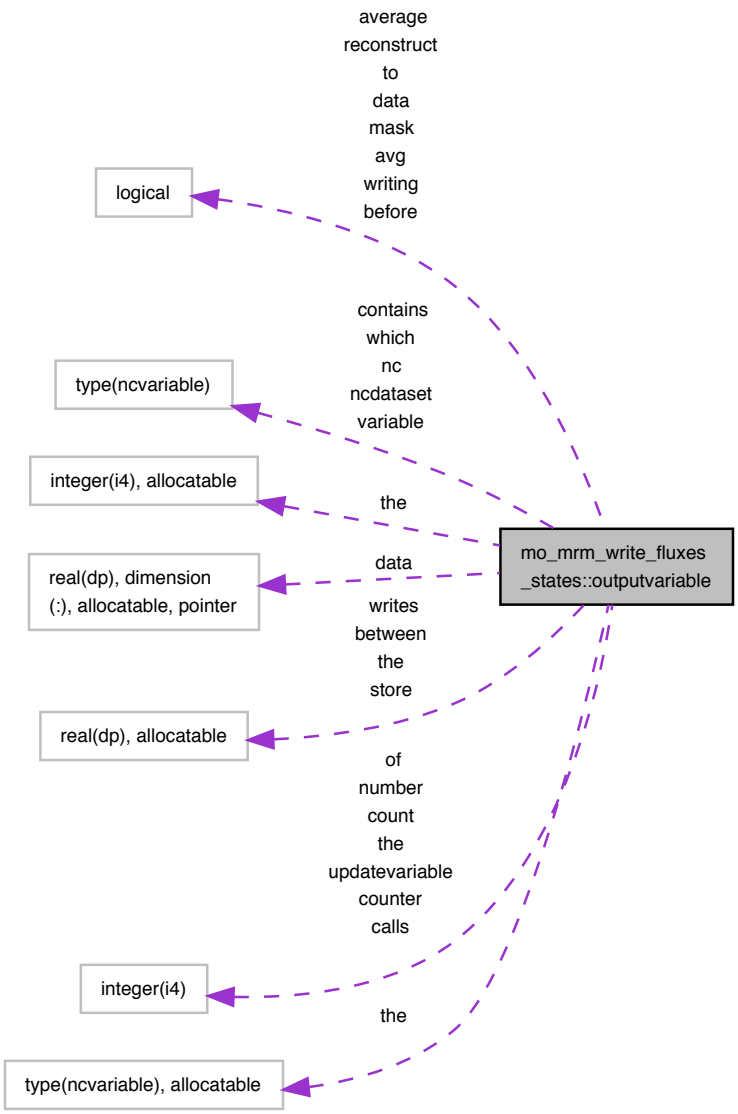
```
integer(i4) mo_write_fluxes_states::outputdataset::written
```

The documentation for this interface was generated from the following file:

- mo_write_fluxes_states.f90

17.83 mo_mrm_write_fluxes_states::outputvariable Interface Reference

Collaboration diagram for mo_mrm_write_fluxes_states::outputvariable:



Public Member Functions

- procedure, public updatevariable
- procedure, public writevariabletimestep

Public Attributes

- type(ncvariable) nc
- type(ncvariable) ncdataset

- type(ncvariable) which
- type(ncvariable) contains
- type(ncvariable), allocatable the
- type(ncvariable) variable
- logical avg = .false.
- logical average
- logical, dimension(:), allocatable, pointer data
- logical before
- logical writing
- logical, dimension(:, :), pointer mask
- logical, pointer to
- logical, pointer reconstruct
- real(dp), dimension(:), allocatable, pointer data
- real(dp), allocatable store
- real(dp), allocatable the
- real(dp), allocatable between
- real(dp), allocatable writes
- integer(i4) counter = 0
- integer(i4) count
- integer(i4), allocatable the
- integer(i4) number
- integer(i4) of
- integer(i4), public updatevariable
- integer(i4) calls

17.83.1 Member Function/Subroutine Documentation

17.83.1.1 updatevariable()

```
procedure, public mo_mrm_write_fluxes_states::outputvariable::updatevariable ( )
```

17.83.1.2 writevariabletimestep()

```
procedure, public mo_mrm_write_fluxes_states::outputvariable::writevariabletimestep ( )
```

17.83.2 Member Data Documentation

17.83.2.1 average

```
logical mo_mrm_write_fluxes_states::outputvariable::average
```

17.83.2.2 avg

```
logical mo_mrm_write_fluxes_states::outputvariable::avg = .false.
```

17.83.2.3 before

```
logical mo_mrm_write_fluxes_states::outputvariable::before
```

17.83.2.4 between

```
real(dp), allocatable mo_mrm_write_fluxes_states::outputvariable::between
```

17.83.2.5 calls

```
integer(i4) mo_mrm_write_fluxes_states::outputvariable::calls
```

17.83.2.6 contains

```
type(ncvariable) mo_mrm_write_fluxes_states::outputvariable::contains
```

17.83.2.7 count

```
integer(i4) mo_mrm_write_fluxes_states::outputvariable::count
```

17.83.2.8 counter

```
integer(i4) mo_mrm_write_fluxes_states::outputvariable::counter = 0
```

17.83.2.9 data [1/2]

```
logical, dimension(:), allocatable, pointer mo_mrm_write_fluxes_states::outputvariable::data
```

17.83.2.10 data [2/2]

```
real(dp), dimension(:), allocatable, pointer mo_mrm_write_fluxes_states::outputvariable::data
```

17.83.2.11 mask

```
logical, dimension(:, :), pointer mo_mrm_write_fluxes_states::outputvariable::mask
```

17.83.2.12 nc

```
type(ncvariable) mo_mrm_write_fluxes_states::outputvariable::nc
```

17.83.2.13 ncdataset

```
type(ncvariable) mo_mrm_write_fluxes_states::outputvariable::ncdataset
```

17.83.2.14 number

```
integer(i4) mo_mrm_write_fluxes_states::outputvariable::number
```

17.83.2.15 of

```
integer(i4) mo_mrm_write_fluxes_states::outputvariable::of
```

17.83.2.16 reconstruct

```
logical, pointer mo_mrm_write_fluxes_states::outputvariable::reconstruct
```

17.83.2.17 store

```
real(dp), allocatable mo_mrm_write_fluxes_states::outputvariable::store
```

17.83.2.18 the [1/3]

```
type(ncvariable), allocatable mo_mrm_write_fluxes_states::outputvariable::the
```

17.83.2.19 the [2/3]

```
real(dp), allocatable mo_mrm_write_fluxes_states::outputvariable::the
```

17.83.2.20 the [3/3]

```
integer(i4), allocatable mo_mrm_write_fluxes_states::outputvariable::the
```

17.83.2.21 to

```
logical, pointer mo_mrm_write_fluxes_states::outputvariable::to
```

17.83.2.22 updatevariable

```
integer(i4), public mo_mrm_write_fluxes_states::outputvariable::updatevariable
```

17.83.2.23 variable

```
type(ncvariable) mo_mrm_write_fluxes_states::outputvariable::variable
```

17.83.2.24 which

```
type(ncvariable) mo_mrm_write_fluxes_states::outputvariable::which
```

17.83.2.25 writes

```
real(dp), allocatable mo_mrm_write_fluxes_states::outputvariable::writes
```

17.83.2.26 writing

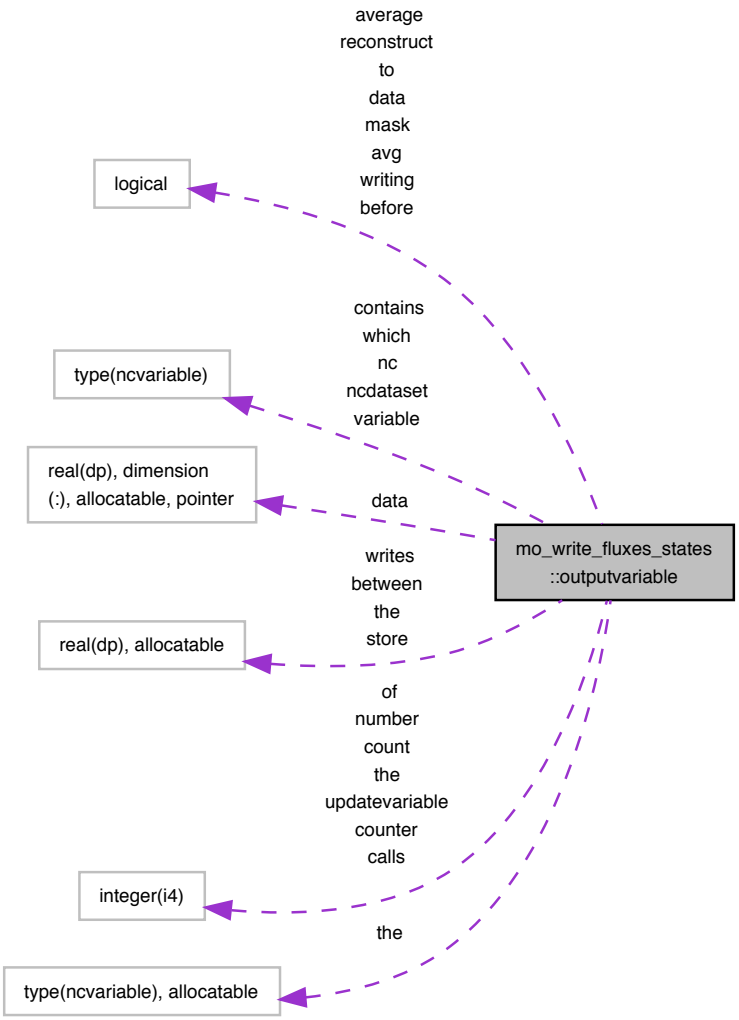
```
logical mo_mrm_write_fluxes_states::outputvariable::writing
```

The documentation for this interface was generated from the following file:

- mo_mrm_write_fluxes_states.f90

17.84 mo_write_fluxes_states::outputvariable Interface Reference

Collaboration diagram for mo_write_fluxes_states::outputvariable:



Public Member Functions

- procedure, public updatevariable
- procedure, public writevariabletimestep

Public Attributes

- type(ncvariable) nc
- type(ncvariable) ncdataset
- type(ncvariable) which
- type(ncvariable) contains
- type(ncvariable), allocatable the

- type(ncvariable) variable
- logical avg = .false.
- logical average
- logical, dimension(:), allocatable, pointer data
- logical before
- logical writing
- logical, dimension(:, :), pointer mask
- logical, pointer to
- logical, pointer reconstruct
- real(dp), dimension(:), allocatable, pointer data
- real(dp), allocatable store
- real(dp), allocatable the
- real(dp), allocatable between
- real(dp), allocatable writes
- integer(i4) counter = 0
- integer(i4) count
- integer(i4), allocatable the
- integer(i4) number
- integer(i4) of
- integer(i4), public updatevariable
- integer(i4) calls

17.84.1 Member Function/Subroutine Documentation

17.84.1.1 updatevariable()

```
procedure, public mo_write_fluxes_states::outputvariable::updatevariable ( )
```

17.84.1.2 writevariabletimestep()

```
procedure, public mo_write_fluxes_states::outputvariable::writevariabletimestep ( )
```

17.84.2 Member Data Documentation

17.84.2.1 average

```
logical mo_write_fluxes_states::outputvariable::average
```

17.84.2.2 avg

```
logical mo_write_fluxes_states::outputvariable::avg = .false.
```

17.84.2.3 before

```
logical mo_write_fluxes_states::outputvariable::before
```

17.84.2.4 between

```
real(dp), allocatable mo_write_fluxes_states::outputvariable::between
```

17.84.2.5 calls

```
integer(i4) mo_write_fluxes_states::outputvariable::calls
```

17.84.2.6 contains

```
type(ncvariable) mo_write_fluxes_states::outputvariable::contains
```

17.84.2.7 count

```
integer(i4) mo_write_fluxes_states::outputvariable::count
```

17.84.2.8 counter

```
integer(i4) mo_write_fluxes_states::outputvariable::counter = 0
```

17.84.2.9 data [1/2]

```
logical, dimension(:), allocatable, pointer mo_write_fluxes_states::outputvariable::data
```

17.84.2.10 data [2/2]

```
real(dp), dimension(:), allocatable, pointer mo_write_fluxes_states::outputvariable::data
```

17.84.2.11 mask

```
logical, dimension(:, :), pointer mo_write_fluxes_states::outputvariable::mask
```

17.84.2.12 nc

type(ncvariable) mo_write_fluxes_states::outputvariable::nc

17.84.2.13 ncdataset

type(ncvariable) mo_write_fluxes_states::outputvariable::ncdataset

17.84.2.14 number

integer(i4) mo_write_fluxes_states::outputvariable::number

17.84.2.15 of

integer(i4) mo_write_fluxes_states::outputvariable::of

17.84.2.16 reconstruct

logical, pointer mo_write_fluxes_states::outputvariable::reconstruct

17.84.2.17 store

real(dp), allocatable mo_write_fluxes_states::outputvariable::store

17.84.2.18 the [1/3]

type(ncvariable), allocatable mo_write_fluxes_states::outputvariable::the

17.84.2.19 the [2/3]

real(dp), allocatable mo_write_fluxes_states::outputvariable::the

17.84.2.20 the [3/3]

integer(i4), allocatable mo_write_fluxes_states::outputvariable::the

17.84.2.21 to

```
logical, pointer mo_write_fluxes_states::outputvariable::to
```

17.84.2.22 updatevariable

```
integer(i4), public mo_write_fluxes_states::outputvariable::updatevariable
```

17.84.2.23 variable

```
type(ncvariable) mo_write_fluxes_states::outputvariable::variable
```

17.84.2.24 which

```
type(ncvariable) mo_write_fluxes_states::outputvariable::which
```

17.84.2.25 writes

```
real(dp), allocatable mo_write_fluxes_states::outputvariable::writes
```

17.84.2.26 writing

```
logical mo_write_fluxes_states::outputvariable::writing
```

The documentation for this interface was generated from the following file:

- mo_write_fluxes_states.f90

17.85 mo_append::paste Interface Reference

Paste (columns) scalars, vectors, and matrixes onto existing array.

Public Member Functions

- subroutine paste_i4_m_s (mat1, sca2)
- subroutine paste_i4_m_v (mat1, vec2, fill_value)
- subroutine paste_i4_m_m (mat1, mat2, fill_value)
- subroutine paste_i8_m_s (mat1, sca2)
- subroutine paste_i8_m_v (mat1, vec2, fill_value)
- subroutine paste_i8_m_m (mat1, mat2, fill_value)
- subroutine paste_sp_m_s (mat1, sca2)
- subroutine paste_sp_m_v (mat1, vec2, fill_value)
- subroutine paste_sp_m_m (mat1, mat2, fill_value)
- subroutine paste_dp_m_s (mat1, sca2)

- subroutine paste_dp_m_v (mat1, vec2, fill_value)
- subroutine paste_dp_m_m (mat1, mat2, fill_value)
- subroutine paste_char_m_s (mat1, sca2)
- subroutine paste_char_m_v (mat1, vec2, fill_value)
- subroutine paste_char_m_m (mat1, mat2, fill_value)
- subroutine paste_lgt_m_s (mat1, sca2)
- subroutine paste_lgt_m_v (mat1, vec2)
- subroutine paste_lgt_m_m (mat1, mat2)

17.85.1 Detailed Description

Paste (columns) scalars, vectors, and matrixes onto existing array.

Pastes one input to the columns of another, i.e. append on the second dimension.
The input might be a scalar, a vector or a matrix.
Possibilities are:
(1) paste scalar to one-line matrix
(3) paste vector to a matrix
(5) paste matrix to matrix

Parameters

`in`	*input2*	values to paste. Can be INTEGER(I4/I8), REAL(SP/DP), or CHARACTER(len=∗) and also scalar, DIMENSION(:), or DIMENSION(:,:) If not scalar then the rows have to agree with input1
`in,out`	*allocatable :: input1*	array to be pasted to. Can be INTEGER(I4/I8), REAL(SP/DP), or CHARACTER(len=∗). Must be DIMENSION(:) or DIMENSION(:,:), and allocatable. If input2 is not scalar then it must be size(input1,1) = size(input2,1).

Author

Juliane Mai

Date

Aug 2012

17.85.2 Member Function/Subroutine Documentation

17.85.2.1 paste_char_m_m()

```
subroutine mo_append::paste::paste_char_m_m (
             character(len = *), dimension(:, :), intent(inout), allocatable mat1,
             character(len = *), dimension(:, :), intent(in) mat2,
             character(len = *), intent(in), optional fill_value )
```

17.85.2.2 paste_char_m_s()

```
subroutine mo_append::paste::paste_char_m_s (
             character(len = *), dimension(:, :), intent(inout), allocatable mat1,
             character(len = *), intent(in) sca2 )
```

17.85.2.3 paste_char_m_v()

```
subroutine mo_append::paste::paste_char_m_v (
             character(len = *), dimension(:, :), intent(inout), allocatable mat1,
             character(len = *), dimension(:), intent(in) vec2,
             character(len = *), intent(in), optional fill_value )
```

17.85.2.4 paste_dp_m_m()

```
subroutine mo_append::paste::paste_dp_m_m (
             real(dp), dimension(:, :), intent(inout), allocatable mat1,
             real(dp), dimension(:, :), intent(in) mat2,
             real(dp), intent(in), optional fill_value )
```

17.85.2.5 paste_dp_m_s()

```
subroutine mo_append::paste::paste_dp_m_s (
             real(dp), dimension(:, :), intent(inout), allocatable mat1,
             real(dp), intent(in) sca2 )
```

17.85.2.6 paste_dp_m_v()

```
subroutine mo_append::paste::paste_dp_m_v (
             real(dp), dimension(:, :), intent(inout), allocatable mat1,
             real(dp), dimension(:), intent(in) vec2,
             real(dp), intent(in), optional fill_value )
```

17.85.2.7 paste_i4_m_m()

```
subroutine mo_append::paste::paste_i4_m_m (
             integer(i4), dimension(:, :), intent(inout), allocatable mat1,
             integer(i4), dimension(:, :), intent(in) mat2,
             integer(i4), intent(in), optional fill_value )
```

17.85.2.8 paste_i4_m_s()

```
subroutine mo_append::paste::paste_i4_m_s (
             integer(i4), dimension(:, :), intent(inout), allocatable mat1,
             integer(i4), intent(in) sca2 )
```

17.85.2.9 paste_i4_m_v()

```
subroutine mo_append::paste::paste_i4_m_v (
             integer(i4), dimension(:, :), intent(inout), allocatable mat1,
             integer(i4), dimension(:), intent(in) vec2,
             integer(i4), intent(in), optional fill_value )
```

17.85.2.10 paste_i8_m_m()

```
subroutine mo_append::paste::paste_i8_m_m (
             integer(i8), dimension(:, :), intent(inout), allocatable mat1,
             integer(i8), dimension(:, :), intent(in) mat2,
             integer(i8), intent(in), optional fill_value )
```

17.85.2.11 paste_i8_m_s()

```
subroutine mo_append::paste::paste_i8_m_s (
             integer(i8), dimension(:, :), intent(inout), allocatable mat1,
             integer(i8), intent(in) sca2 )
```

17.85.2.12 paste_i8_m_v()

```
subroutine mo_append::paste::paste_i8_m_v (
             integer(i8), dimension(:, :), intent(inout), allocatable mat1,
             integer(i8), dimension(:), intent(in) vec2,
             integer(i8), intent(in), optional fill_value )
```

17.85.2.13 paste_lgt_m_m()

```
subroutine mo_append::paste::paste_lgt_m_m (
             logical, dimension(:, :), intent(inout), allocatable mat1,
             logical, dimension(:, :), intent(in) mat2 )
```

17.85.2.14 paste_lgt_m_s()

```
subroutine mo_append::paste::paste_lgt_m_s (
             logical, dimension(:, :), intent(inout), allocatable mat1,
             logical, intent(in) sca2 )
```

17.85.2.15 paste_lgt_m_v()

```
subroutine mo_append::paste::paste_lgt_m_v (
             logical, dimension(:, :), intent(inout), allocatable mat1,
             logical, dimension(:), intent(in) vec2 )
```

17.85.2.16 paste_sp_m_m()

```
subroutine mo_append::paste::paste_sp_m_m (
             real(sp), dimension(:, :), intent(inout), allocatable mat1,
             real(sp), dimension(:, :), intent(in) mat2,
             real(sp), intent(in), optional fill_value )
```

17.85.2.17 paste_sp_m_s()

```
subroutine mo_append::paste::paste_sp_m_s (
             real(sp), dimension(:, :), intent(inout), allocatable mat1,
             real(sp), intent(in) sca2 )
```

17.85.2.18 paste_sp_m_v()

```
subroutine mo_append::paste::paste_sp_m_v (
             real(sp), dimension(:, :), intent(inout), allocatable mat1,
             real(sp), dimension(:), intent(in) vec2,
             real(sp), intent(in), optional fill_value )
```

The documentation for this interface was generated from the following file:

- mo_append.f90

17.86 mo_spatialsimilarity::pd Interface Reference

Calculates pattern dissimilarity (PD) measure.

Public Member Functions

- real(sp) function pd_sp (mat1, mat2, mask, valid)
- real(dp) function pd_dp (mat1, mat2, mask, valid)

17.86.1 Detailed Description

Calculates pattern dissimilarity (PD) measure.

PD = 1 - sum(dissimilarity(mat1, mat2)) / count(mask) dissimilarity(mat1, mat2) = comparison if pixel is larger than its neighbouring values

An array element value is compared with its 8 neighbouring cells to check if these cells are larger than the array element value. The result is a 3x3 matrix in which larger cells are indicated by a true value. For comparison this is done with both input arrays. The resulting array is the sum of xor values of the 3x3 matrices for each of the both arrays. This means only neighbourhood comparisons which are different in the 2 matrices are counted. This resulting matrix is afterwards normalized to its available neighbors. Furthermore an average over the entire field is calculated. The valid interval of the values for PD is [0..1]. In which 1 indicates full agreement and 0 full dismatching.

```
EXAMPLE:~
```

```
mat1 =  | 12 17  1 | , mat2 = |  7  9 12 |
        |  4 10 11 |          | 12 11 11 |
        | 15  2 20 |          |  5 13  7 |

booleans determined for every grid cell following fortran array scrolling
i.e. (/col1_row1, col1_row2, col1_row3, col2_row1, .. ,col3_row3/),(/3,3/)

comp1 = | FFF FFF FTF, FFF FFF FFF, FTT FFT FFF |
        | FFF TFT TTF, TFT TFF FTT, TFF FFT FFF |
        | FFF FFF FFF, TTF TFF TTF, FFF FFF FFF |

comp2 = | FFF FFT FTT, FFT FFT FTT, FFF FFF FFF |
        | FFF FFF FFT, FTF FFT TFF, FFT TFF FFF |
        | FFF TFF TTF, FFF FFF FFF, TTF TFF FFF |

xor=neq = | FFF FFT FFT, FFT FFT FTT, FTT FFT FFF |
          | FFF TFT TTT, TTT TFT TTT, TFT TFT FFF |
          | FFF TFF TTF, TTF TFF TTF, TTF TFF FFF |

            DISSIMILAR / VALID NEIGH CELLS
PDMatrix = | 2, 4, 3 | / | 3, 5, 3 | = | 0.66, 0.80, 1.00 |
           | 5, 8, 4 |   | 5, 8, 5 |   | 1.00, 1.00, 0.80 |
           | 3, 5, 3 |   | 3, 5, 3 |   | 1.00, 1.00, 1.00 |

PD = 1 - sum(PDMatrix) / count(mask) = 1 - (8.2666666 / 9) = 0.08148
```

If an optinal mask is given, the calculations are over those locations that correspond to true values in the mask. mat1 and mat2 can be single or double precision. The result will have the same numerical precision.

Parameters

in	*real(sp/dp), dimension(:,:) :: mat1*	2D-array with input numbers
in	*real(sp/dp), dimension(:,:) :: mat2*	2D-array with input numbers
in	*logical,dimension(:,:),optinal :: mask*	2D-array of logical values with size(mat1/mat2) If present, only those locations in mat1/mat2 having true values in mask are evaluated.
out	*logical,optinal :: valid*	indicates if the function could determine a valid value result can be unvalid if entire mask is .false. for ex. in this case PD is set to 0 (worst case)

Returns

real(sp/dp) :: PD — pattern dissimilarity measure

Author

Matthias Zink and Juliane Mai

Date

Jan 2013

17.86.2 Member Function/Subroutine Documentation

17.86.2.1 pd_dp()

```
real(dp) function mo_spatialsimilarity::pd::pd_dp (
             real(dp), dimension(:, :), intent(in) mat1,
             real(dp), dimension(:, :), intent(in) mat2,
             logical, dimension(:, :), intent(in), optional mask,
             logical, intent(out), optional valid )
```

17.86.2.2 pd_sp()

```
real(sp) function mo_spatialsimilarity::pd::pd_sp (
             real(sp), dimension(:, :), intent(in) mat1,
             real(sp), dimension(:, :), intent(in) mat2,
             logical, dimension(:, :), intent(in), optional mask,
             logical, intent(out), optional valid )
```

The documentation for this interface was generated from the following file:

- mo_spatialsimilarity.f90

17.87 mo_percentile::percentile Interface Reference

Public Member Functions

- real(sp) function percentile_0d_sp (arrin, k, mask, mode_in)
- real(dp) function percentile_0d_dp (arrin, k, mask, mode_in)
- real(sp) function, dimension(size(k)) percentile_1d_sp (arrin, k, mask, mode_in)
- real(dp) function, dimension(size(k)) percentile_1d_dp (arrin, k, mask, mode_in)

17.87.1 Member Function/Subroutine Documentation

17.87.1.1 percentile_0d_dp()

```
real(dp) function mo_percentile::percentile::percentile_0d_dp (
             real(dp), dimension(:), intent(in) arrin,
             real(dp), intent(in) k,
             logical, dimension(:), intent(in), optional mask,
             integer(i4), intent(in), optional mode_in )
```

17.87.1.2 percentile_0d_sp()

```
real(sp) function mo_percentile::percentile::percentile_0d_sp (
             real(sp), dimension(:), intent(in) arrin,
             real(sp), intent(in) k,
             logical, dimension(:), intent(in), optional mask,
             integer(i4), intent(in), optional mode_in )
```

17.87.1.3 percentile_1d_dp()

```
real(dp) function, dimension(size(k)) mo_percentile::percentile::percentile_1d_dp (
            real(dp), dimension(:), intent(in) arrin,
            real(dp), dimension(:), intent(in) k,
            logical, dimension(:), intent(in), optional mask,
            integer(i4), intent(in), optional mode_in )
```

17.87.1.4 percentile_1d_sp()

```
real(sp) function, dimension(size(k)) mo_percentile::percentile::percentile_1d_sp (
            real(sp), dimension(:), intent(in) arrin,
            real(sp), dimension(:), intent(in) k,
            logical, dimension(:), intent(in), optional mask,
            integer(i4), intent(in), optional mode_in )
```

The documentation for this interface was generated from the following file:

- mo_percentile.f90

17.88 mo_common_variables::period Type Reference

Collaboration diagram for mo_common_variables::period:

Public Attributes

- integer(i4) dstart
- integer(i4) mstart
- integer(i4) ystart
- integer(i4) dend

- integer(i4) mend
- integer(i4) yend
- integer(i4) julstart
- integer(i4) julend
- integer(i4) nobs

17.88.1 Member Data Documentation

17.88.1.1 dend

```
integer(i4) mo_common_variables::period::dend
```

17.88.1.2 dstart

```
integer(i4) mo_common_variables::period::dstart
```

17.88.1.3 julend

```
integer(i4) mo_common_variables::period::julend
```

17.88.1.4 julstart

```
integer(i4) mo_common_variables::period::julstart
```

17.88.1.5 mend

```
integer(i4) mo_common_variables::period::mend
```

17.88.1.6 mstart

```
integer(i4) mo_common_variables::period::mstart
```

17.88.1.7 nobs

```
integer(i4) mo_common_variables::period::nobs
```

17.88.1.8 yend

```
integer(i4) mo_common_variables::period::yend
```

17.88.1.9 ystart

```
integer(i4) mo_common_variables::period::ystart
```

The documentation for this type was generated from the following file:

- mo_common_variables.f90

17.89 mo_percentile::qmedian Interface Reference

Public Member Functions

- real(sp) function qmedian_sp (dat)
- real(dp) function qmedian_dp (dat)

17.89.1 Member Function/Subroutine Documentation

17.89.1.1 qmedian_dp()

```
real(dp) function mo_percentile::qmedian::qmedian_dp (
             real(dp), dimension(:), intent(inout) dat )
```

17.89.1.2 qmedian_sp()

```
real(sp) function mo_percentile::qmedian::qmedian_sp (
             real(sp), dimension(:), intent(inout) dat )
```

The documentation for this interface was generated from the following file:

- mo_percentile.f90

17.90 mo_orderpack::rapknr Interface Reference

Public Member Functions

- subroutine d_rapknr (XDONT, IRNGT, NORD)
- subroutine r_rapknr (XDONT, IRNGT, NORD)
- subroutine i_rapknr (XDONT, IRNGT, NORD)

17.90.1 Member Function/Subroutine Documentation

17.90.1.1 d_rapknr()

```
subroutine mo_orderpack::rapknr::d_rapknr (
             real(kind = dp), dimension (:), intent(in) XDONT,
             integer(kind = i4), dimension (:), intent(out) IRNGT,
             integer(kind = i4), intent(in) NORD )
```

17.90.1.2 i_rapknr()

```
subroutine mo_orderpack::rapknr::i_rapknr (
             integer(kind = i4), dimension (:), intent(in) XDONT,
             integer(kind = i4), dimension (:), intent(out) IRNGT,
             integer(kind = i4), intent(in) NORD )
```

17.90.1.3 r_rapknr()

```
subroutine mo_orderpack::rapknr::r_rapknr (
             real(kind = sp), dimension (:), intent(in) XDONT,
             integer(kind = i4), dimension (:), intent(out) IRNGT,
             integer(kind = i4), intent(in) NORD )
```

The documentation for this interface was generated from the following file:

- mo_orderpack.f90

17.91 mo_read_spatial_data::read_spatial_data_ascii Interface Reference

Reads spatial data files of ASCII format.

Public Member Functions

- subroutine read_spatial_data_ascii_i4 (filename, fileunit, header_ncols, header_nrows, header_xllcorner, header_yllcorner, header_cellsize, data, mask)

  TODO: add description.
- subroutine read_spatial_data_ascii_dp (filename, fileunit, header_ncols, header_nrows, header_xllcorner, header_yllcorner, header_cellsize, data, mask)

  TODO: add description.

17.91.1 Detailed Description

Reads spatial data files of ASCII format.

Reads spatial input data, e.g. dem, aspect, flow direction.

Authors

Juliane Mai

Date

Jan 2013

17.91.2 Member Function/Subroutine Documentation

17.91.2.1 read_spatial_data_ascii_dp()

```
subroutine mo_read_spatial_data::read_spatial_data_ascii::read_spatial_data_ascii_dp (
            character(len = *), intent(in) filename,
            integer(i4), intent(in) fileunit,
            integer(i4), intent(in) header_ncols,
            integer(i4), intent(in) header_nrows,
            real(dp), intent(in) header_xllcorner,
            real(dp), intent(in) header_yllcorner,
            real(dp), intent(in) header_cellsize,
            real(dp), dimension(:, :), intent(out), allocatable data,
            logical, dimension(:, :), intent(out), allocatable mask )
```

TODO: add description.

TODO: add description

Parameters

in	*character(len = *) :: filename*	filename with location
in	*integer(i4) :: fileunit*	unit for opening the file
in	*integer(i4) :: header_nCols*	number of columns of data fields:
in	*integer(i4) :: header_nRows*	number of rows of data fields:
in	*real(dp) :: header_xllcorner*	header read in lower left corner
in	*real(dp) :: header_yllcorner*	header read in lower left corner
in	*real(dp) :: header_cellsize*	header read in cellsize
out	*real(dp), dimension(:, :) :: data*	data
out	*logical, dimension(:, :) :: mask*	mask

Authors

Robert Schweppe

Date

Jun 2018

17.91.2.2 read_spatial_data_ascii_i4()

```
subroutine mo_read_spatial_data::read_spatial_data_ascii::read_spatial_data_ascii_i4 (
            character(len = *), intent(in) filename,
            integer(i4), intent(in) fileunit,
            integer(i4), intent(in) header_ncols,
            integer(i4), intent(in) header_nrows,
            real(dp), intent(in) header_xllcorner,
            real(dp), intent(in) header_yllcorner,
            real(dp), intent(in) header_cellsize,
            integer(i4), dimension(:, :), intent(out), allocatable data,
            logical, dimension(:, :), intent(out), allocatable mask )
```

TODO: add description.

TODO: add description

Parameters

in	*character(len = ∗) :: filename*	filename with location
in	*integer(i4) :: fileunit*	unit for opening the file
in	*integer(i4) :: header_nCols*	number of columns of data fields:
in	*integer(i4) :: header_nRows*	number of rows of data fields:
in	*real(dp) :: header_xllcorner*	header read in lower left corner
in	*real(dp) :: header_yllcorner*	header read in lower left corner
in	*real(dp) :: header_cellsize*	header read in cellsize
out	*integer(i4), dimension(:, :) :: data*	data
out	*logical, dimension(:, :) :: mask*	mask

Authors

Robert Schweppe

Date

Jun 2018

The documentation for this interface was generated from the following file:

- mo_read_spatial_data.f90

17.92 mo_corr::realft Interface Reference

Private Member Functions

- subroutine realft_sp (data, isign, zdata)
- subroutine realft_dp (data, isign, zdata)

17.92.1 Member Function/Subroutine Documentation

17.92.1.1 realft_dp()

```
subroutine mo_corr::realft::realft_dp (
            real(dp), dimension(:), intent(inout) data,
            integer(i4), intent(in) isign,
            complex(dpc), dimension(:), optional, target zdata )  [private]
```

17.92.1.2 realft_sp()

```
subroutine mo_corr::realft::realft_sp (
            real(sp), dimension(:), intent(inout) data,
            integer(i4), intent(in) isign,
            complex(spc), dimension(:), optional, target zdata )  [private]
```

The documentation for this interface was generated from the following file:

- mo_corr.f90

17.93 mo_orderpack::refpar Interface Reference

Public Member Functions

- subroutine d_refpar (XDONT, IRNGT, NORD)
- subroutine r_refpar (XDONT, IRNGT, NORD)
- subroutine i_refpar (XDONT, IRNGT, NORD)

17.93.1 Member Function/Subroutine Documentation

17.93.1.1 d_refpar()

```
subroutine mo_orderpack::refpar::d_refpar (
            real(kind = dp), dimension (:), intent(in) XDONT,
            integer(kind = i4), dimension (:), intent(out) IRNGT,
            integer(kind = i4), intent(in) NORD )
```

17.93.1.2 i_refpar()

```
subroutine mo_orderpack::refpar::i_refpar (
            integer(kind = i4), dimension (:), intent(in) XDONT,
            integer(kind = i4), dimension (:), intent(out) IRNGT,
            integer(kind = i4), intent(in) NORD )
```

17.93.1.3 r_refpar()

```
subroutine mo_orderpack::refpar::r_refpar (
            real(kind = sp), dimension (:), intent(in) XDONT,
            integer(kind = i4), dimension (:), intent(out) IRNGT,
            integer(kind = i4), intent(in) NORD )
```

The documentation for this interface was generated from the following file:

- mo_orderpack.f90

17.94 mo_orderpack::refsor Interface Reference

Public Member Functions

- subroutine d_refsor (XDONT)
- subroutine r_refsor (XDONT)
- subroutine i_refsor (XDONT)

17.94.1 Member Function/Subroutine Documentation

17.94.1.1 d_refsor()

```
subroutine mo_orderpack::refsor::d_refsor (
             real(kind = dp), dimension (:), intent(inout) XDONT )
```

17.94.1.2 i_refsor()

```
subroutine mo_orderpack::refsor::i_refsor (
             integer(kind = i4), dimension (:), intent(inout) XDONT )
```

17.94.1.3 r_refsor()

```
subroutine mo_orderpack::refsor::r_refsor (
             real(kind = sp), dimension (:), intent(inout) XDONT )
```

The documentation for this interface was generated from the following file:

- mo_orderpack.f90

17.95 mo_orderpack::rinpar Interface Reference

Public Member Functions

- subroutine d_rinpar (XDONT, IRNGT, NORD)
- subroutine r_rinpar (XDONT, IRNGT, NORD)
- subroutine i_rinpar (XDONT, IRNGT, NORD)

17.95.1 Member Function/Subroutine Documentation

17.95.1.1 d_rinpar()

```
subroutine mo_orderpack::rinpar::d_rinpar (
             real(kind = dp), dimension (:), intent(in) XDONT,
```

```
            integer(kind = i4), dimension (:), intent(out) IRNGT,
            integer(kind = i4), intent(in) NORD )
```

17.95.1.2 i_rinpar()

```
subroutine mo_orderpack::rinpar::i_rinpar (
            integer(kind = i4), dimension (:), intent(in) XDONT,
            integer(kind = i4), dimension (:), intent(out) IRNGT,
            integer(kind = i4), intent(in) NORD )
```

17.95.1.3 r_rinpar()

```
subroutine mo_orderpack::rinpar::r_rinpar (
            real(kind = sp), dimension (:), intent(in) XDONT,
            integer(kind = i4), dimension (:), intent(out) IRNGT,
            integer(kind = i4), intent(in) NORD )
```

The documentation for this interface was generated from the following file:

- mo_orderpack.f90

17.96 mo_errormeasures::rmse Interface Reference

Public Member Functions

- real(sp) function rmse_sp_1d (x, y, mask)
- real(dp) function rmse_dp_1d (x, y, mask)
- real(sp) function rmse_sp_2d (x, y, mask)
- real(dp) function rmse_dp_2d (x, y, mask)
- real(sp) function rmse_sp_3d (x, y, mask)
- real(dp) function rmse_dp_3d (x, y, mask)

17.96.1 Member Function/Subroutine Documentation

17.96.1.1 rmse_dp_1d()

```
real(dp) function mo_errormeasures::rmse::rmse_dp_1d (
            real(dp), dimension(:), intent(in) x,
            real(dp), dimension(:), intent(in) y,
            logical, dimension(:), intent(in), optional mask )
```

17.96.1.2 rmse_dp_2d()

```
real(dp) function mo_errormeasures::rmse::rmse_dp_2d (
            real(dp), dimension(:, :), intent(in) x,
```

```
            real(dp), dimension(:, :), intent(in) y,
            logical, dimension(:, :), intent(in), optional mask )
```

17.96.1.3 rmse_dp_3d()

```
real(dp) function mo_errormeasures::rmse::rmse_dp_3d (
            real(dp), dimension(:, :, :), intent(in) x,
            real(dp), dimension(:, :, :), intent(in) y,
            logical, dimension(:, :, :), intent(in), optional mask )
```

17.96.1.4 rmse_sp_1d()

```
real(sp) function mo_errormeasures::rmse::rmse_sp_1d (
            real(sp), dimension(:), intent(in) x,
            real(sp), dimension(:), intent(in) y,
            logical, dimension(:), intent(in), optional mask )
```

17.96.1.5 rmse_sp_2d()

```
real(sp) function mo_errormeasures::rmse::rmse_sp_2d (
            real(sp), dimension(:, :), intent(in) x,
            real(sp), dimension(:, :), intent(in) y,
            logical, dimension(:, :), intent(in), optional mask )
```

17.96.1.6 rmse_sp_3d()

```
real(sp) function mo_errormeasures::rmse::rmse_sp_3d (
            real(sp), dimension(:, :, :), intent(in) x,
            real(sp), dimension(:, :, :), intent(in) y,
            logical, dimension(:, :, :), intent(in), optional mask )
```

The documentation for this interface was generated from the following file:

- mo_errormeasures.f90

17.97 mo_orderpack::rnkpar Interface Reference

Public Member Functions

- subroutine d_rnkpar (XDONT, IRNGT, NORD)
- subroutine r_rnkpar (XDONT, IRNGT, NORD)
- subroutine i_rnkpar (XDONT, IRNGT, NORD)

17.97.1 Member Function/Subroutine Documentation

17.97.1.1 d_rnkpar()

```
subroutine mo_orderpack::rnkpar::d_rnkpar (
             real(kind = dp), dimension (:), intent(in) XDONT,
             integer(kind = i4), dimension (:), intent(out) IRNGT,
             integer(kind = i4), intent(in) NORD )
```

17.97.1.2 i_rnkpar()

```
subroutine mo_orderpack::rnkpar::i_rnkpar (
             integer(kind = i4), dimension (:), intent(in) XDONT,
             integer(kind = i4), dimension (:), intent(out) IRNGT,
             integer(kind = i4), intent(in) NORD )
```

17.97.1.3 r_rnkpar()

```
subroutine mo_orderpack::rnkpar::r_rnkpar (
             real(kind = sp), dimension (:), intent(in) XDONT,
             integer(kind = i4), dimension (:), intent(out) IRNGT,
             integer(kind = i4), intent(in) NORD )
```

The documentation for this interface was generated from the following file:

- mo_orderpack.f90

17.98 mo_errormeasures::sae Interface Reference

Public Member Functions

- real(sp) function sae_sp_1d (x, y, mask)
- real(dp) function sae_dp_1d (x, y, mask)
- real(sp) function sae_sp_2d (x, y, mask)
- real(dp) function sae_dp_2d (x, y, mask)
- real(sp) function sae_sp_3d (x, y, mask)
- real(dp) function sae_dp_3d (x, y, mask)

17.98.1 Member Function/Subroutine Documentation

17.98.1.1 sae_dp_1d()

```
real(dp) function mo_errormeasures::sae::sae_dp_1d (
             real(dp), dimension(:), intent(in) x,
             real(dp), dimension(:), intent(in) y,
             logical, dimension(:), intent(in), optional mask )
```

17.98.1.2 sae_dp_2d()

```
real(dp) function mo_errormeasures::sae::sae_dp_2d (
             real(dp), dimension(:, :), intent(in) x,
             real(dp), dimension(:, :), intent(in) y,
             logical, dimension(:, :), intent(in), optional mask )
```

17.98.1.3 sae_dp_3d()

```
real(dp) function mo_errormeasures::sae::sae_dp_3d (
             real(dp), dimension(:, :, :), intent(in) x,
             real(dp), dimension(:, :, :), intent(in) y,
             logical, dimension(:, :, :), intent(in), optional mask )
```

17.98.1.4 sae_sp_1d()

```
real(sp) function mo_errormeasures::sae::sae_sp_1d (
             real(sp), dimension(:), intent(in) x,
             real(sp), dimension(:), intent(in) y,
             logical, dimension(:), intent(in), optional mask )
```

17.98.1.5 sae_sp_2d()

```
real(sp) function mo_errormeasures::sae::sae_sp_2d (
             real(sp), dimension(:, :), intent(in) x,
             real(sp), dimension(:, :), intent(in) y,
             logical, dimension(:, :), intent(in), optional mask )
```

17.98.1.6 sae_sp_3d()

```
real(sp) function mo_errormeasures::sae::sae_sp_3d (
             real(sp), dimension(:, :, :), intent(in) x,
             real(sp), dimension(:, :, :), intent(in) y,
             logical, dimension(:, :, :), intent(in), optional mask )
```

The documentation for this interface was generated from the following file:

- mo_errormeasures.f90

17.99 mo_julian::setcalendar Interface Reference

Private Member Functions

- subroutine setcalendarinteger (selector)

  Set module private variable calendar.
- subroutine setcalendarstring (selector)

  Set module private variable calendar.

17.99.1 Member Function/Subroutine Documentation

17.99.1.1 setcalendarinteger()

```
subroutine mo_julian::setcalendar::setcalendarinteger (
            integer(i4), intent(in) selector )  [private]
```

Set module private variable calendar.

Parameters

in	*integer(i4) :: selector*	{1|2|3}

Author

Written, David Schaefer

Date

Jan 2015

17.99.1.2 setcalendarstring()

```
subroutine mo_julian::setcalendar::setcalendarstring (
            character(*), intent(in) selector )  [private]
```

Set module private variable calendar.

Parameters

in	*character(len=*) :: selector*	{"julian"|"365day"|"360day"}

Author

Written, David Schaefer

Date

Jan 2015

The documentation for this interface was generated from the following file:

- mo_julian.f90

17.100 mo_moment::skewness Interface Reference

Public Member Functions

- real(sp) function skewness_sp (dat, mask)
- real(dp) function skewness_dp (dat, mask)

17.100.1 Member Function/Subroutine Documentation

17.100.1.1 skewness_dp()

```
real(dp) function mo_moment::skewness::skewness_dp (
             real(dp), dimension(:), intent(in) dat,
             logical, dimension(:), intent(in), optional mask )
```

17.100.1.2 skewness_sp()

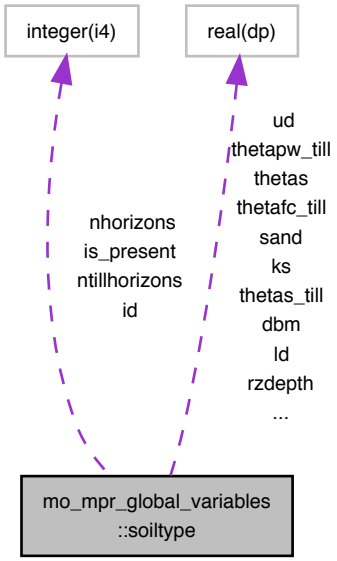
```
real(sp) function mo_moment::skewness::skewness_sp (
             real(sp), dimension(:), intent(in) dat,
             logical, dimension(:), intent(in), optional mask )
```

The documentation for this interface was generated from the following file:

- mo_moment.f90

17.101 mo_mpr_global_variables::soiltype Type Reference

Collaboration diagram for mo_mpr_global_variables::soiltype:



Private Attributes

- integer(i4), dimension(:), allocatable id

- integer(i4), dimension(:), allocatable nhorizons
- integer(i4), dimension(:), allocatable is_present
- real(dp), dimension(:, :), allocatable ud
- real(dp), dimension(:, :), allocatable ld
- real(dp), dimension(:, :), allocatable clay
- real(dp), dimension(:, :), allocatable sand
- real(dp), dimension(:, :), allocatable dbm
- real(dp), dimension(:, :), allocatable depth
- real(dp), dimension(:), allocatable rzdepth
- real(dp), dimension(:, :, :), allocatable wd
- integer(i4), dimension(:), allocatable ntillhorizons
- real(dp), dimension(:, :, :), allocatable thetas_till
- real(dp), dimension(:, :), allocatable thetas
- real(dp), dimension(:, :, :), allocatable db
- real(dp), dimension(:, :, :), allocatable thetafc_till
- real(dp), dimension(:, :), allocatable thetafc
- real(dp), dimension(:, :, :), allocatable thetapw_till
- real(dp), dimension(:, :), allocatable thetapw
- real(dp), dimension(:, :, :), allocatable ks

17.101.1 Member Data Documentation

17.101.1.1 clay

```
real(dp), dimension(:, :), allocatable mo_mpr_global_variables::soiltype::clay  [private]
```

17.101.1.2 db

```
real(dp), dimension(:, :, :), allocatable mo_mpr_global_variables::soiltype::db  [private]
```

17.101.1.3 dbm

```
real(dp), dimension(:, :), allocatable mo_mpr_global_variables::soiltype::dbm  [private]
```

17.101.1.4 depth

```
real(dp), dimension(:, :), allocatable mo_mpr_global_variables::soiltype::depth  [private]
```

17.101.1.5 id

```
integer(i4), dimension(:), allocatable mo_mpr_global_variables::soiltype::id  [private]
```

17.101.1.6 is_present

```
integer(i4), dimension(:), allocatable mo_mpr_global_variables::soiltype::is_present  [private]
```

17.101.1.7 ks

```
real(dp), dimension(:, :, :), allocatable mo_mpr_global_variables::soiltype::ks  [private]
```

17.101.1.8 ld

```
real(dp), dimension(:, :), allocatable mo_mpr_global_variables::soiltype::ld  [private]
```

17.101.1.9 nhorizons

```
integer(i4), dimension(:), allocatable mo_mpr_global_variables::soiltype::nhorizons  [private]
```

17.101.1.10 ntillhorizons

```
integer(i4), dimension(:), allocatable mo_mpr_global_variables::soiltype::ntillhorizons  [private]
```

17.101.1.11 rzdepth

```
real(dp), dimension(:), allocatable mo_mpr_global_variables::soiltype::rzdepth  [private]
```

17.101.1.12 sand

```
real(dp), dimension(:, :), allocatable mo_mpr_global_variables::soiltype::sand  [private]
```

17.101.1.13 thetafc

```
real(dp), dimension(:, :), allocatable mo_mpr_global_variables::soiltype::thetafc  [private]
```

17.101.1.14 thetafc_till

```
real(dp), dimension(:, :, :), allocatable mo_mpr_global_variables::soiltype::thetafc_till
[private]
```

17.101.1.15 thetapw

```
real(dp), dimension(:, :), allocatable mo_mpr_global_variables::soiltype::thetapw  [private]
```

17.101.1.16 thetapw_till

```
real(dp), dimension(:, :, :), allocatable mo_mpr_global_variables::soiltype::thetapw_till
[private]
```

17.101.1.17 thetas

```
real(dp), dimension(:, :), allocatable mo_mpr_global_variables::soiltype::thetas  [private]
```

17.101.1.18 thetas_till

```
real(dp), dimension(:, :, :), allocatable mo_mpr_global_variables::soiltype::thetas_till  [private]
```

17.101.1.19 ud

```
real(dp), dimension(:, :), allocatable mo_mpr_global_variables::soiltype::ud  [private]
```

17.101.1.20 wd

```
real(dp), dimension(:, :, :), allocatable mo_mpr_global_variables::soiltype::wd  [private]
```

The documentation for this type was generated from the following file:

- mo_mpr_global_variables.f90

17.102 mo_orderpack::sort Interface Reference

Unconditional ranking.

Public Member Functions

- subroutine d_refsor (XDONT)
- subroutine r_refsor (XDONT)
- subroutine i_refsor (XDONT)

17.102.1 Detailed Description

Unconditional ranking.

Subroutine MRGRNK (XVALT, IMULT) Ranks array XVALT into index array IRNGT, using merge-sort For performance reasons, the first 2 passes are taken out of the standard loop, and use dedicated coding.

Subroutine MRGREF (XVALT, IRNGT) Ranks array XVALT into index array IRNGT, using merge-sort This version is not optimized for performance, and is thus not as difficult to read as the previous one.

Partial ranking

Subroutine RNKPAR (XVALT, IRNGT, NORD) Ranks partially XVALT by IRNGT, up to order NORD (refined for speed) This routine uses a pivoting strategy such as the one of finding the median based on the quicksort algorithm, but we skew the pivot choice to try to bring it to NORD as fast as possible. It uses 2 temporary arrays, one where it stores the indices of the values smaller than the pivot, and the other for the indices of values larger than the pivot that we might still need later on. It iterates until it can bring the number of values in ILOWT to exactly NORD, and then uses an insertion sort to rank this set, since it is supposedly small.

Subroutine RAPKNR (XVALT, IRNGT, NORD) Same as RNKPAR, but in decreasing order (RAPKNR = RNKPAR spelt backwards).

Subroutine REFPAR (XVALT, IRNGT, NORD) Ranks partially XVALT by IRNGT, up to order NORD This version is not optimized for performance, and is thus not as difficult to read as some other ones. It uses a pivoting strategy such as the one of finding the median based on the quicksort algorithm. It uses a temporary array, where it stores the partially ranked indices of the values. It iterates until it can bring the number of values lower than the pivot to exactly NORD, and then uses an insertion sort to rank this set, since it is supposedly small.

Subroutine RINPAR (XVALT, IRNGT, NORD) Ranks partially XVALT by IRNGT, up to order NORD This version is not optimized for performance, and is thus not as difficult to read as some other ones. It uses insertion sort, limiting insertion to the first NORD values. It does not use any work array and is faster when NORD is very small (2-5), but worst case behavior (intially inverse sorted) can easily happen. In many cases, the refined quicksort method is faster.

Integer Function INDNTH (XVALT, NORD) Returns the index of the NORDth value of XVALT (in increasing order) This routine uses a pivoting strategy such as the one of finding the median based on the quicksort algorithm, but we skew the pivot choice to try to bring it to NORD as fast as possible. It uses 2 temporary arrays, one where it stores the indices of the values smaller than the pivot, and the other for the indices of values larger than the pivot that we might still need later on. It iterates until it can bring the number of values in ILOWT to exactly NORD, and then takes out the original index of the maximum value in this set.

Subroutine INDMED (XVALT, INDM) Returns the index of the median (((Size(XVALT)+1))/2th value) of XVALT This routine uses the recursive procedure described in Knuth, The Art of Computer Programming, vol. 3, 5.3.3 - This procedure is linear in time, and does not require to be able to interpolate in the set as the one used in INDNTH. It also has better worst case behavior than INDNTH, but is about 10% slower in average for random uniformly distributed values.

Note that in Orderpack 1.0, this routine was a Function procedure, and is now changed to a Subroutine.

Unique ranking

Subroutine UNIRNK (XVALT, IRNGT, NUNI) Ranks an array, removing duplicate entries (uses merge sort). The routine is similar to pure merge-sort ranking, but on the last pass, it discards indices that correspond to duplicate entries. For performance reasons, the first 2 passes are taken out of the standard loop, and use dedicated coding.

Subroutine UNIPAR (XVALT, IRNGT, NORD) Ranks partially XVALT by IRNGT, up to order NORD at most, removing duplicate entries This routine uses a pivoting strategy such as the one of finding the median based on the quicksort algorithm, but we skew the pivot choice to try to bring it to NORD as quickly as possible. It uses 2 temporary arrays, one where it stores the indices of the values smaller than the pivot, and the other for the indices of values larger than the pivot that we might still need later on. It iterates until it can bring the number of values in ILOWT to exactly NORD, and then uses an insertion sort to rank this set, since it is supposedly small. At all times, the NORD first values in ILOWT correspond to distinct values of the input array.

Subroutine UNIINV (XVALT, IGOEST) Inverse ranking of an array, with removal of duplicate entries The routine is similar to pure merge-sort ranking, but on the last pass, it sets indices in IGOEST to the rank of the original value in an ordered set with duplicates removed. For performance reasons, the first 2 passes are taken out of the standard loop, and use dedicated coding.

Subroutine MULCNT (XVALT, IMULT) Gives, for each array value, its multiplicity The number of times that a value appears in the array is computed by using inverse ranking, counting for each rank the number of values that “collide"

to this rank, and returning this sum to the locations in the original set. Uses subroutine UNIINV.

Random permutation: an interesting use of ranking

A variation of the following problem was raised on the internet sci.math.num-analysis news group: Given an array, I would like to find a random permutation of this array that I could control with a `nearbyness'' parameter so that elements stay close to their initial locations. The`nearbyness" parameter ranges from 0 to 1, with 0 such that no element moves from its initial location, and 1 such that the permutation is fully random.

Subroutine CTRPER (XVALT, PCLS) Permute array XVALT randomly, but leaving elements close to their initial locations The routine takes the 1...size(XVALT) index array as real values, takes a combination of these values and of random values as a perturbation of the index array, and sorts the initial set according to the ranks of these perturbated indices. The relative proportion of initial order and random order is 1-PCLS / PCLS, thus when PCLS = 0, there is no change in the order whereas the new order is fully random when PCLS = 1. Uses subroutine MRGRNK.

The above solution found another application when I was asked the following question: I am given two arrays, representing parents' incomes and their children's incomes, but I do not know which parents correspond to which children. I know from an independent source the value of the correlation coefficient between the incomes of the parents and of their children. I would like to pair the elements of these arrays so that the given correlation coefficient is attained, i.e. to reconstruct a realistic dataset, though very likely not to be the true one.

Program GIVCOR Given two arrays of equal length of unordered values, find a "matching value" in the second array for each value in the first so that the global correlation coefficient reaches exactly a given target The routine first sorts the two arrays, so as to get the match of maximum possible correlation. It then iterates, applying the random permutation algorithm of controlled disorder ctrper to the second array. When the resulting correlation goes beyond (lower than) the target correlation, one steps back and reduces the disorder parameter of the permutation. When the resulting correlation lies between the current one and the target, one replaces the array with the newly permuted one. When the resulting correlation increases from the current value, one increases the disorder parameter. That way, the target correlation is approached from above, by a controlled increase in randomness. Since full randomness leads to zero correlation, the iterations meet the desired coefficient at some point. It may be noted that there could be some cases when one would get stuck in a sort of local minimum, where local perturbations cannot further reduce the correlation and where global ones lead to overpass the target. It seems easier to restart the program with a different seed when this occurs than to design an avoidance scheme. Also, should a negative correlation be desired, the program should be modified to start with one array in reverse order with respect to the other, i.e. coorelation as close to -1 as possible.

Sorting

Full sorting

Subroutine INSSOR (XVALT) Sorts XVALT into increasing order (Insertion sort) This subroutine uses insertion sort. It does not use any work array and is faster when XVALT is of very small size (< 20), or already almost sorted, but worst case behavior (intially inverse sorted) can easily happen. In most cases, the quicksort or merge sort method is faster.

Subroutine REFSOR (XVALT) Sorts XVALT into increasing order (Quick sort) This version is not optimized for performance, and is thus not as difficult to read as some other ones. This subroutine uses quicksort in a recursive implementation, and insertion sort for the last steps with small subsets. It does not use any work array

Partial sorting

Subroutine INSPAR (XVALT, NORD) Sorts partially XVALT, bringing the NORD lowest values at the begining of the array. This subroutine uses insertion sort, limiting insertion to the first NORD values. It does not use any work array and is faster when NORD is very small (2-5), but worst case behavior can happen fairly probably (initially inverse sorted). In many cases, the refined quicksort method is faster.

Function FNDNTH (XVALT, NORD) Finds out and returns the NORDth value in XVALT (ascending order) This subroutine uses insertion sort, limiting insertion to the first NORD values, and even less when one can know that the value that is considered will not be the NORDth. It uses only a work array of size NORD and is faster when NORD is very small (2-5), but worst case behavior can happen fairly probably (initially inverse sorted). In many cases, the refined quicksort method implemented by VALNTH / INDNTH is faster, though much more difficult to read and understand.

Function VALNTH (XVALT, NORD) Finds out and returns the NORDth value in XVALT (ascending order) This subroutine simply calls INDNTH.

Function VALMED (XVALT) Finds out and returns the median (((Size(XVALT)+1))/2th value) of XVALT This routine uses the recursive procedure described in Knuth, The Art of Computer Programming, vol. 3, 5.3.3 - This procedure is linear in time, and does not require to be able to interpolate in the set as the one used in VALNTH/INDNTH. It also has better worst case behavior than VALNTH/INDNTH, and is about 20% faster in average for random uniformly distributed values.

Function OMEDIAN (XVALT) It is a modified version of VALMED that provides the average between the two middle values in the case Size(XVALT) is even.

Unique sorting

Subroutine UNISTA (XVALT, NUNI) Removes duplicates from an array This subroutine uses merge sort unique inverse ranking. It leaves in the initial set only those entries that are unique, packing the array, and leaving the order of the retained values unchanged.

17.102.2 Member Function/Subroutine Documentation

17.102.2.1 d_refsor()

```
subroutine mo_orderpack::sort::d_refsor (
             real(kind = dp), dimension (:), intent(inout) XDONT )
```

17.102.2.2 i_refsor()

```
subroutine mo_orderpack::sort::i_refsor (
             integer(kind = i4), dimension (:), intent(inout) XDONT )
```

17.102.2.3 r_refsor()

```
subroutine mo_orderpack::sort::r_refsor (
             real(kind = sp), dimension (:), intent(inout) XDONT )
```

The documentation for this interface was generated from the following file:

- mo_orderpack.f90

17.103 mo_orderpack::sort_index Interface Reference

Public Member Functions

- integer(i4) function, dimension(size(arr)) sort_index_dp (arr)
- integer(i4) function, dimension(size(arr)) sort_index_sp (arr)
- integer(i4) function, dimension(size(arr)) sort_index_i4 (arr)

17.103.1 Member Function/Subroutine Documentation

17.103.1.1 sort_index_dp()

```
integer(i4) function, dimension(size(arr)) mo_orderpack::sort_index::sort_index_dp (
             real(dp), dimension(:), intent(in) arr )
```

17.103.1.2 sort_index_i4()

```
integer(i4) function, dimension(size(arr)) mo_orderpack::sort_index::sort_index_i4 (
             integer(i4), dimension(:), intent(in) arr )
```

17.103.1.3 sort_index_sp()

```
integer(i4) function, dimension(size(arr)) mo_orderpack::sort_index::sort_index_sp (
             real(sp), dimension(:), intent(in) arr )
```

The documentation for this interface was generated from the following file:

- mo_orderpack.f90

17.104 mo_spatial_agg_disagg_forcing::spatial_aggregation Interface Reference

Spatial aggregation of meterological variables.

Public Member Functions

- subroutine spatial_aggregation_3d (data2, cellsize2, cellsize1, mask1, mask2, data1)
- subroutine spatial_aggregation_4d (data2, cellsize2, cellsize1, mask1, mask2, data1)

17.104.1 Detailed Description

Spatial aggregation of meterological variables.

Aggregate (or upscale) the given level-2 meteorological data to the required level-1 spatial resolution for the mHM run.

Authors

Rohini Kumar

Date

Jan 2013

17.104.2 Member Function/Subroutine Documentation

17.104.2.1 spatial_aggregation_3d()

```
subroutine mo_spatial_agg_disagg_forcing::spatial_aggregation::spatial_aggregation_3d (
             real(dp), dimension(:, :, :), intent(in) data2,
             real(dp), intent(in) cellsize2,
             real(dp), intent(in) cellsize1,
             logical, dimension(:, :), intent(in) mask1,
             logical, dimension(:, :), intent(in) mask2,
             real(dp), dimension(:, :, :), intent(out), allocatable data1 )
```

17.104.2.2 spatial_aggregation_4d()

```
subroutine mo_spatial_agg_disagg_forcing::spatial_aggregation::spatial_aggregation_4d (
             real(dp), dimension(:, :, :, :), intent(in) data2,
             real(dp), intent(in) cellsize2,
             real(dp), intent(in) cellsize1,
             logical, dimension(:, :), intent(in) mask1,
             logical, dimension(:, :), intent(in) mask2,
             real(dp), dimension(:, :, :, :), intent(out), allocatable data1 )
```

The documentation for this interface was generated from the following file:

- mo_spatial_agg_disagg_forcing.f90

17.105 mo_spatial_agg_disagg_forcing::spatial_disaggregation Interface Reference

Spatial disaggregation of meterological variables.

Public Member Functions

- subroutine spatial_disaggregation_3d (data2, cellsize2, cellsize1, mask1, mask2, data1)
- subroutine spatial_disaggregation_4d (data2, cellsize2, cellsize1, mask1, mask2, data1)

17.105.1 Detailed Description

Spatial disaggregation of meterological variables.

Disaggregate (or downscale) the given level-2 meteorological data to the required level-1 spatial resolution for the mHM run.

Parameters

in	*real(dp), dimension(:, :, :) :: data2*	Level-2 data
in	*real(dp) :: cellsize2*	Level-2 resolution
in	*real(dp) :: cellsize1*	Level-1 resolution
in	*logical, dimension(:, :) :: mask1*	Level-1 mask
in	*logical, dimension(:, :) :: mask2*	Level-2 mask
out	*real(dp), dimension(:, :, :) :: data1*	Level-1 data

Authors

Rohini Kumar

Date

Jan 2013

17.105.2 Member Function/Subroutine Documentation

17.105.2.1 spatial_disaggregation_3d()

```
subroutine mo_spatial_agg_disagg_forcing::spatial_disaggregation::spatial_disaggregation_3d (
             real(dp), dimension(:, :, :), intent(in) data2,
             real(dp), intent(in) cellsize2,
             real(dp), intent(in) cellsize1,
             logical, dimension(:, :), intent(in) mask1,
             logical, dimension(:, :), intent(in) mask2,
             real(dp), dimension(:, :, :), intent(out), allocatable data1 )
```

17.105.2.2 spatial_disaggregation_4d()

```
subroutine mo_spatial_agg_disagg_forcing::spatial_disaggregation::spatial_disaggregation_4d (
             real(dp), dimension(:, :, :, :), intent(in) data2,
             real(dp), intent(in) cellsize2,
             real(dp), intent(in) cellsize1,
             logical, dimension(:, :), intent(in) mask1,
             logical, dimension(:, :), intent(in) mask2,
             real(dp), dimension(:, :, :, :), intent(out), allocatable data1 )
```

The documentation for this interface was generated from the following file:

- mo_spatial_agg_disagg_forcing.f90

17.106 mo_utils::special_value Interface Reference

Special IEEE values.

Public Member Functions

- real(sp) function special_value_sp (x, ieee)
- real(dp) function special_value_dp (x, ieee)

17.106.1 Detailed Description

Special IEEE values.

Returns special IEEE values such as Infinity or Not-a-Number.
Wraps to function ieee_value of the intrinsic module ieee_arithmetic but gives alternatives for gfortran, which does not provide ieee_arithmetic.

Quiet and signaling NaN are the same in case of gfortran;
also denormal values are the same as inf.

Current special values are:
IEEE_SIGNALING_NAN
IEEE_QUIET_NAN
IEEE_NEGATIVE_INF
IEEE_POSITIVE_INF
IEEE_NEGATIVE_DENORMAL
IEEE_POSITIVE_DENORMAL
IEEE_NEGATIVE_NORMAL
IEEE_POSITIVE_NORMAL
IEEE_NEGATIVE_ZERO
IEEE_POSITIVE_ZERO

Parameters

in	*real(sp/dp) :: x*	dummy for kind of output
in	*"character(le=*)*	:: name ieee signal nanme

Returns

real(sp/dp) :: special_value — IEEE special value
IEEE_SIGNALING_NAN
IEEE_QUIET_NAN (==IEEE_SIGNALING_NAN for gfortran)
IEEE_NEGATIVE_INF
IEEE_POSITIVE_INF
IEEE_NEGATIVE_DENORMAL (==-0.0 for gfortran)
IEEE_POSITIVE_DENORMAL (==0.0 for gfortran)
IEEE_NEGATIVE_NORMAL (==-1.0 for gfortran)
IEEE_POSITIVE_NORMAL (==1.0 for gfortran)
IEEE_NEGATIVE_ZERO
IEEE_POSITIVE_ZERO

Authors

Matthias Cuntz

Date

Mar 2015

17.106.2 Member Function/Subroutine Documentation

17.106.2.1 special_value_dp()

```
real(dp) function mo_utils::special_value::special_value_dp (
             real(dp), intent(in) x,
             character(len = *), intent(in) ieee )
```

17.106.2.2 special_value_sp()

```
real(sp) function mo_utils::special_value::special_value_sp (
            real(sp), intent(in) x,
            character(len = *), intent(in) ieee )
```

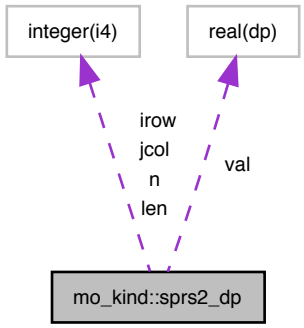
The documentation for this interface was generated from the following file:

- mo_utils.f90

17.107 mo_kind::sprs2_dp Type Reference

Double Precision Numerical Recipes types for sparse arrays.

Collaboration diagram for mo_kind::sprs2_dp:



Public Attributes

- integer(i4) n
- integer(i4) len
- real(dp), dimension(:), pointer val
- integer(i4), dimension(:), pointer irow
- integer(i4), dimension(:), pointer jcol

17.107.1 Detailed Description

Double Precision Numerical Recipes types for sparse arrays.

17.107.2 Member Data Documentation

17.107.2.1 irow

```
integer(i4), dimension(:), pointer mo_kind::sprs2_dp::irow
```

17.107.2.2 jcol

```
integer(i4), dimension(:), pointer mo_kind::sprs2_dp::jcol
```

17.107.2.3 len

```
integer(i4) mo_kind::sprs2_dp::len
```

17.107.2.4 n

```
integer(i4) mo_kind::sprs2_dp::n
```

17.107.2.5 val

```
real(dp), dimension(:), pointer mo_kind::sprs2_dp::val
```

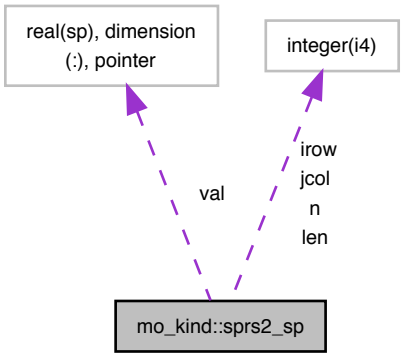
The documentation for this type was generated from the following file:

- mo_kind.f90

17.108 mo_kind::sprs2_sp Type Reference

Single Precision Numerical Recipes types for sparse arrays.

Collaboration diagram for mo_kind::sprs2_sp:



Public Attributes

- integer(i4) n
- integer(i4) len

- real(sp), dimension(:), pointer val
- integer(i4), dimension(:), pointer irow
- integer(i4), dimension(:), pointer jcol

17.108.1 Detailed Description

Single Precision Numerical Recipes types for sparse arrays.

17.108.2 Member Data Documentation

17.108.2.1 irow

```
integer(i4), dimension(:), pointer mo_kind::sprs2_sp::irow
```

17.108.2.2 jcol

```
integer(i4), dimension(:), pointer mo_kind::sprs2_sp::jcol
```

17.108.2.3 len

```
integer(i4) mo_kind::sprs2_sp::len
```

17.108.2.4 n

```
integer(i4) mo_kind::sprs2_sp::n
```

17.108.2.5 val

```
real(sp), dimension(:), pointer mo_kind::sprs2_sp::val
```

The documentation for this type was generated from the following file:

- mo_kind.f90

17.109 mo_errormeasures::sse Interface Reference

Public Member Functions

- real(sp) function sse_sp_1d (x, y, mask)
- real(dp) function sse_dp_1d (x, y, mask)
- real(sp) function sse_sp_2d (x, y, mask)
- real(dp) function sse_dp_2d (x, y, mask)
- real(sp) function sse_sp_3d (x, y, mask)
- real(dp) function sse_dp_3d (x, y, mask)

17.109.1 Member Function/Subroutine Documentation

17.109.1.1 sse_dp_1d()

```
real(dp) function mo_errormeasures::sse::sse_dp_1d (
             real(dp), dimension(:), intent(in) x,
             real(dp), dimension(:), intent(in) y,
             logical, dimension(:), intent(in), optional mask )
```

17.109.1.2 sse_dp_2d()

```
real(dp) function mo_errormeasures::sse::sse_dp_2d (
             real(dp), dimension(:, :), intent(in) x,
             real(dp), dimension(:, :), intent(in) y,
             logical, dimension(:, :), intent(in), optional mask )
```

17.109.1.3 sse_dp_3d()

```
real(dp) function mo_errormeasures::sse::sse_dp_3d (
             real(dp), dimension(:, :, :), intent(in) x,
             real(dp), dimension(:, :, :), intent(in) y,
             logical, dimension(:, :, :), intent(in), optional mask )
```

17.109.1.4 sse_sp_1d()

```
real(sp) function mo_errormeasures::sse::sse_sp_1d (
             real(sp), dimension(:), intent(in) x,
             real(sp), dimension(:), intent(in) y,
             logical, dimension(:), intent(in), optional mask )
```

17.109.1.5 sse_sp_2d()

```
real(sp) function mo_errormeasures::sse::sse_sp_2d (
             real(sp), dimension(:, :), intent(in) x,
             real(sp), dimension(:, :), intent(in) y,
             logical, dimension(:, :), intent(in), optional mask )
```

17.109.1.6 sse_sp_3d()

```
real(sp) function mo_errormeasures::sse::sse_sp_3d (
             real(sp), dimension(:, :, :), intent(in) x,
             real(sp), dimension(:, :, :), intent(in) y,
             logical, dimension(:, :, :), intent(in), optional mask )
```

The documentation for this interface was generated from the following file:

- mo_errormeasures.f90

17.110 mo_standard_score::standard_score Interface Reference

Calculates the standard score / normalization (anomaly) / z-score.

Public Member Functions

- real(sp) function, dimension(size(data, dim=1)) standard_score_sp (data, mask)
- real(dp) function, dimension(size(data, dim=1)) standard_score_dp (data, mask)

17.110.1 Detailed Description

Calculates the standard score / normalization (anomaly) / z-score.

In statistics, the standard score is the (signed) number of standard deviations an observation or datum is above the mean. Thus, a positive standard score indicates a datum above the mean, while a negative standard score indicates a datum below the mean. It is a dimensionless quantity obtained by subtracting the population mean from an individual raw score and then dividing the difference by the population standard deviation. This conversion process is called standardizing or normalizing (however, "normalizing" can refer to many types of ratios).
Standard scores are also called z-values, z-scores, normal scores, and standardized variables; the use of "Z" is because the normal distribution is also known as the "Z distribution". They are most frequently used to compare a sample to a standard normal deviate, though they can be defined without assumptions of normality (Wikipedia, May 2015).

$$standard\_score = \frac{x - \mu_x}{\sigma_x}$$

where μ_x is the mean of a population x and σ_x its standard deviation.

If an optinal mask is given, the calculations are over those locations that correspond to true values in the mask. data can be single or double precision. The result will have the same numerical precision.

Parameters

in	*real(sp/dp), dimension(:) :: data*	data to calculate the standard score for
in	*logical, dimension(:),optinal :: mask*	indication which cells to use for calculation If present, only those locations in mask having true values in mask are evaluated.

Returns

real(sp/dp) :: standard_score — standard score / normalization (anomaly) / z-score

Note

Richard J. Larsen and Morris L. Marx (2000) An Introduction to Mathematical Statistics and Its Applications, Third Edition, ISBN 0-13-922303-7. p. 282.

Author

Matthias Zink

Date

May 2015

17.110.2 Member Function/Subroutine Documentation

17.110.2.1 standard_score_dp()

```
real(dp) function, dimension(size(data, dim = 1)) mo_standard_score::standard_score::standard↩
_score_dp (
             real(dp), dimension(:), intent(in) data,
             logical, dimension(:), intent(in), optional mask )
```

17.110.2.2 standard_score_sp()

```
real(sp) function, dimension(size(data, dim = 1)) mo_standard_score::standard_score::standard↩
_score_sp (
             real(sp), dimension(:), intent(in) data,
             logical, dimension(:), intent(in), optional mask )
```

The documentation for this interface was generated from the following file:

- mo_standard_score.f90

17.111 mo_moment::stddev Interface Reference

Public Member Functions

- real(sp) function stddev_sp (dat, mask)
- real(dp) function stddev_dp (dat, mask)

17.111.1 Member Function/Subroutine Documentation

17.111.1.1 stddev_dp()

```
real(dp) function mo_moment::stddev::stddev_dp (
             real(dp), dimension(:), intent(in) dat,
             logical, dimension(:), intent(in), optional mask )
```

17.111.1.2 stddev_sp()

```
real(sp) function mo_moment::stddev::stddev_sp (
             real(sp), dimension(:), intent(in) dat,
             logical, dimension(:), intent(in), optional mask )
```

The documentation for this interface was generated from the following file:

- mo_moment.f90

17.112 mo_corr::swap Interface Reference

Private Member Functions

- subroutine swap_1d_spc (a, b)
- subroutine swap_1d_dpc (a, b)

17.112.1 Member Function/Subroutine Documentation

17.112.1.1 swap_1d_dpc()

```
subroutine mo_corr::swap::swap_1d_dpc (
             complex(dpc), dimension(:), intent(inout) a,
             complex(dpc), dimension(:), intent(inout) b )  [private]
```

17.112.1.2 swap_1d_spc()

```
subroutine mo_corr::swap::swap_1d_spc (
             complex(spc), dimension(:), intent(inout) a,
             complex(spc), dimension(:), intent(inout) b )  [private]
```

The documentation for this interface was generated from the following file:

- mo_corr.f90

17.113 mo_utils::swap Interface Reference

Swap to values or two elements in array.

Public Member Functions

- elemental pure subroutine swap_xy_dp (x, y)
- elemental pure subroutine swap_xy_sp (x, y)
- elemental pure subroutine swap_xy_i4 (x, y)
- subroutine swap_vec_dp (x, i1, i2)
- subroutine swap_vec_sp (x, i1, i2)
- subroutine swap_vec_i4 (x, i1, i2)

17.113.1 Detailed Description

Swap to values or two elements in array.

Swaps either two entities, i.e. scalars, vectors, matrices, or two elements in a vector. The call is either
call swap(x,y)
or
call swap(vec,i,j)

Parameters

in	*integer(i4) :: i*	Index of first element to be swapped with second [case swap(vec,i,j)]

Parameters

in	*integer(i4) :: j*	Index of second element to be swapped with first [case swap(vec,i,j)]
in,out	*real(sp/dp/i4) :: x[(:,...)]*	First scalar or array to swap with second [case swap(x,y)]
in,out	*real(sp/dp/i4) :: y[(:[,:])]*	Second scalar or array to swap with first [case swap(x,y)]
in,out	*real(sp/dp/i4) :: x(:)*	Vector of which to elements are swapped [case swap(vec,i,j)]

Author

Matthias Cuntz

Date

May 2014

17.113.2 Member Function/Subroutine Documentation

17.113.2.1 swap_vec_dp()

```
subroutine mo_utils::swap::swap_vec_dp (
             real(dp), dimension(:), intent(inout) x,
             integer(i4), intent(in) i1,
             integer(i4), intent(in) i2 )
```

17.113.2.2 swap_vec_i4()

```
subroutine mo_utils::swap::swap_vec_i4 (
             integer(i4), dimension(:), intent(inout) x,
             integer(i4), intent(in) i1,
             integer(i4), intent(in) i2 )
```

17.113.2.3 swap_vec_sp()

```
subroutine mo_utils::swap::swap_vec_sp (
             real(sp), dimension(:), intent(inout) x,
             integer(i4), intent(in) i1,
             integer(i4), intent(in) i2 )
```

17.113.2.4 swap_xy_dp()

```
elemental pure subroutine mo_utils::swap::swap_xy_dp (
             real(dp), intent(inout) x,
             real(dp), intent(inout) y )
```

17.113.2.5 swap_xy_i4()

```
elemental pure subroutine mo_utils::swap::swap_xy_i4 (
            integer(i4), intent(inout) x,
            integer(i4), intent(inout) y )
```

17.113.2.6 swap_xy_sp()

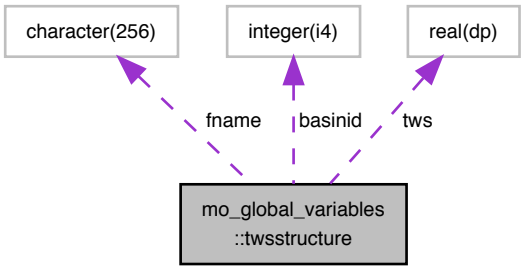
```
elemental pure subroutine mo_utils::swap::swap_xy_sp (
            real(sp), intent(inout) x,
            real(sp), intent(inout) y )
```

The documentation for this interface was generated from the following file:

- mo_utils.f90

17.114 mo_global_variables::twsstructure Type Reference

Collaboration diagram for mo_global_variables::twsstructure:



Public Attributes

- integer(i4), dimension(:), allocatable basinid
- character(256), dimension(:), allocatable fname
- real(dp), dimension(:, :), allocatable tws

17.114.1 Member Data Documentation

17.114.1.1 basinid

```
integer(i4), dimension(:), allocatable mo_global_variables::twsstructure::basinid
```

17.114.1.2 fname

```
character(256), dimension(:), allocatable mo_global_variables::twsstructure::fname
```

17.114.1.3 tws

```
real(dp), dimension(:, :), allocatable mo_global_variables::twsstructure::tws
```

The documentation for this type was generated from the following file:

- mo_global_variables.f90

17.115 mo_orderpack::uniinv Interface Reference

Public Member Functions

- subroutine d_uniinv (XDONT, IGOEST)
- subroutine r_uniinv (XDONT, IGOEST)
- subroutine i_uniinv (XDONT, IGOEST)

17.115.1 Member Function/Subroutine Documentation

17.115.1.1 d_uniinv()

```
subroutine mo_orderpack::uniinv::d_uniinv (
             real(kind = dp), dimension (:), intent(in) XDONT,
             integer(kind = i4), dimension (:), intent(out) IGOEST )
```

17.115.1.2 i_uniinv()

```
subroutine mo_orderpack::uniinv::i_uniinv (
             integer(kind = i4), dimension (:), intent(in) XDONT,
             integer(kind = i4), dimension (:), intent(out) IGOEST )
```

17.115.1.3 r_uniinv()

```
subroutine mo_orderpack::uniinv::r_uniinv (
             real(kind = sp), dimension (:), intent(in) XDONT,
             integer(kind = i4), dimension (:), intent(out) IGOEST )
```

The documentation for this interface was generated from the following file:

- mo_orderpack.f90

17.116 mo_orderpack::unipar Interface Reference

Public Member Functions

- subroutine d_unipar (XDONT, IRNGT, NORD)
- subroutine r_unipar (XDONT, IRNGT, NORD)
- subroutine i_unipar (XDONT, IRNGT, NORD)

17.116.1 Member Function/Subroutine Documentation

17.116.1.1 d_unipar()

```
subroutine mo_orderpack::unipar::d_unipar (
             real(kind = dp), dimension (:), intent(in) XDONT,
             integer(kind = i4), dimension (:), intent(out) IRNGT,
             integer(kind = i4), intent(inout) NORD )
```

17.116.1.2 i_unipar()

```
subroutine mo_orderpack::unipar::i_unipar (
             integer(kind = i4), dimension (:), intent(in) XDONT,
             integer(kind = i4), dimension (:), intent(out) IRNGT,
             integer(kind = i4), intent(inout) NORD )
```

17.116.1.3 r_unipar()

```
subroutine mo_orderpack::unipar::r_unipar (
             real(kind = sp), dimension (:), intent(in) XDONT,
             integer(kind = i4), dimension (:), intent(out) IRNGT,
             integer(kind = i4), intent(inout) NORD )
```

The documentation for this interface was generated from the following file:

- mo_orderpack.f90

17.117 mo_orderpack::unirnk Interface Reference

Public Member Functions

- subroutine d_unirnk (XVALT, IRNGT, NUNI)
- subroutine r_unirnk (XVALT, IRNGT, NUNI)
- subroutine i_unirnk (XVALT, IRNGT, NUNI)

17.117.1 Member Function/Subroutine Documentation

17.117.1.1 d_unirnk()

```
subroutine mo_orderpack::unirnk::d_unirnk (
             real(kind = dp), dimension (:), intent(in) XVALT,
             integer(kind = i4), dimension (:), intent(out) IRNGT,
             integer(kind = i4), intent(out) NUNI )
```

17.117.1.2 i_unirnk()

```
subroutine mo_orderpack::unirnk::i_unirnk (
             integer(kind = i4), dimension (:), intent(in) XVALT,
             integer(kind = i4), dimension (:), intent(out) IRNGT,
             integer(kind = i4), intent(out) NUNI )
```

17.117.1.3 r_unirnk()

```
subroutine mo_orderpack::unirnk::r_unirnk (
             real(kind = sp), dimension (:), intent(in) XVALT,
             integer(kind = i4), dimension (:), intent(out) IRNGT,
             integer(kind = i4), intent(out) NUNI )
```

The documentation for this interface was generated from the following file:

- mo_orderpack.f90

17.118 mo_orderpack::unista Interface Reference

Public Member Functions

- subroutine d_unista (XDONT, NUNI)
- subroutine r_unista (XDONT, NUNI)
- subroutine i_unista (XDONT, NUNI)

17.118.1 Member Function/Subroutine Documentation

17.118.1.1 d_unista()

```
subroutine mo_orderpack::unista::d_unista (
             real(kind = dp), dimension (:), intent(inout) XDONT,
             integer(kind = i4), intent(out) NUNI )
```

17.118.1.2 i_unista()

```
subroutine mo_orderpack::unista::i_unista (
             integer(kind = i4), dimension (:), intent(inout) XDONT,
             integer(kind = i4), intent(out) NUNI )
```

17.118.1.3 r_unista()

```
subroutine mo_orderpack::unista::r_unista (
            real(kind = sp), dimension (:), intent(inout) XDONT,
            integer(kind = i4), intent(out) NUNI )
```

The documentation for this interface was generated from the following file:

- mo_orderpack.f90

17.119 mo_restart::unpack_field_and_write Interface Reference

TODO: add description.

Private Member Functions

- subroutine unpack_field_and_write_1d_i4 (nc, var_name, var_dims, fill_value, data, mask, var_long_name)
- subroutine unpack_field_and_write_1d_dp (nc, var_name, var_dims, fill_value, data, mask, var_long_name)
- subroutine unpack_field_and_write_2d_dp (nc, var_name, var_dims, fill_value, data, mask, var_long_name)
- subroutine unpack_field_and_write_3d_dp (nc, var_name, var_dims, fill_value, data, mask, var_long_name)

17.119.1 Detailed Description

TODO: add description.

TODO: add description

Parameters

in,out	*type(NcDataset) :: nc*	NcDataset to add variable to
in	*character(∗) :: var_name*	variable name
in	*type(NcDimension), dimension(:) :: var_dims*	vector of Variable dimensions
in	*integer(i4) :: fill_value*	fill value used for missing values
in	*integer(i4), dimension(:) :: data*	packed data to be set to variable
in	*logical, dimension(:, :) :: mask*	mask used for unpacking
in	*character(∗), optional :: var_long_name*	variable long name attribute

Authors

Robert Schweppe

Date

Jun 2018

17.119.2 Member Function/Subroutine Documentation

17.119.2.1 unpack_field_and_write_1d_dp()

```
subroutine mo_restart::unpack_field_and_write::unpack_field_and_write_1d_dp (
            type(ncdataset), intent(inout) nc,
```

```
            character(*), intent(in) var_name,
            type(ncdimension), dimension(:), intent(in) var_dims,
            real(dp), intent(in) fill_value,
            real(dp), dimension(:), intent(in) data,
            logical, dimension(:, :), intent(in) mask,
            character(*), intent(in), optional var_long_name )  [private]
```

17.119.2.2 unpack_field_and_write_1d_i4()

```
subroutine mo_restart::unpack_field_and_write::unpack_field_and_write_1d_i4 (
            type(ncdataset), intent(inout) nc,
            character(*), intent(in) var_name,
            type(ncdimension), dimension(:), intent(in) var_dims,
            integer(i4), intent(in) fill_value,
            integer(i4), dimension(:), intent(in) data,
            logical, dimension(:, :), intent(in) mask,
            character(*), intent(in), optional var_long_name )  [private]
```

17.119.2.3 unpack_field_and_write_2d_dp()

```
subroutine mo_restart::unpack_field_and_write::unpack_field_and_write_2d_dp (
            type(ncdataset), intent(inout) nc,
            character(*), intent(in) var_name,
            type(ncdimension), dimension(:), intent(in) var_dims,
            real(dp), intent(in) fill_value,
            real(dp), dimension(:, :), intent(in) data,
            logical, dimension(:, :), intent(in) mask,
            character(*), intent(in), optional var_long_name )  [private]
```

17.119.2.4 unpack_field_and_write_3d_dp()

```
subroutine mo_restart::unpack_field_and_write::unpack_field_and_write_3d_dp (
            type(ncdataset), intent(inout) nc,
            character(*), intent(in) var_name,
            type(ncdimension), dimension(:), intent(in) var_dims,
            real(dp), intent(in) fill_value,
            real(dp), dimension(:, :, :), intent(in) data,
            logical, dimension(:, :), intent(in) mask,
            character(*), intent(in), optional var_long_name )  [private]
```

The documentation for this interface was generated from the following file:

- mo_restart.f90

17.120 mo_mpr_restart::unpack_field_and_write Interface Reference

TODO: add description.

Private Member Functions

- subroutine unpack_field_and_write_1d_i4 (nc, var_name, var_dims, fill_value, data, mask, var_long_name)
- subroutine unpack_field_and_write_1d_dp (nc, var_name, var_dims, fill_value, data, mask, var_long_name)
- subroutine unpack_field_and_write_2d_dp (nc, var_name, var_dims, fill_value, data, mask, var_long_name)
- subroutine unpack_field_and_write_3d_dp (nc, var_name, var_dims, fill_value, data, mask, var_long_name)

17.120.1 Detailed Description

TODO: add description.

TODO: add description

Parameters

in,out	*type(NcDataset) :: nc*	NcDataset to add variable to
in	*character(∗) :: var_name*	variable name
in	*type(NcDimension), dimension(:) :: var_dims*	vector of Variable dimensions
in	*integer(i4) :: fill_value*	fill value used for missing values
in	*integer(i4), dimension(:) :: data*	packed data to be set to variable
in	*logical, dimension(:, :) :: mask*	mask used for unpacking
in	*character(∗), optional :: var_long_name*	variable long name attribute

Authors

Robert Schweppe

Date

Jun 2018

17.120.2 Member Function/Subroutine Documentation

17.120.2.1 unpack_field_and_write_1d_dp()

```
subroutine mo_mpr_restart::unpack_field_and_write::unpack_field_and_write_1d_dp (
             type(ncdataset), intent(inout) nc,
             character(*), intent(in) var_name,
             type(ncdimension), dimension(:), intent(in) var_dims,
             real(dp), intent(in) fill_value,
             real(dp), dimension(:), intent(in) data,
             logical, dimension(:, :), intent(in) mask,
             character(*), intent(in), optional var_long_name )  [private]
```

17.120.2.2 unpack_field_and_write_1d_i4()

```
subroutine mo_mpr_restart::unpack_field_and_write::unpack_field_and_write_1d_i4 (
             type(ncdataset), intent(inout) nc,
             character(*), intent(in) var_name,
```

```
            type(ncdimension), dimension(:), intent(in) var_dims,
            integer(i4), intent(in) fill_value,
            integer(i4), dimension(:), intent(in) data,
            logical, dimension(:, :), intent(in) mask,
            character(*), intent(in), optional var_long_name )  [private]
```

17.120.2.3 unpack_field_and_write_2d_dp()

```
subroutine mo_mpr_restart::unpack_field_and_write::unpack_field_and_write_2d_dp (
            type(ncdataset), intent(inout) nc,
            character(*), intent(in) var_name,
            type(ncdimension), dimension(:), intent(in) var_dims,
            real(dp), intent(in) fill_value,
            real(dp), dimension(:, :), intent(in) data,
            logical, dimension(:, :), intent(in) mask,
            character(*), intent(in), optional var_long_name )  [private]
```

17.120.2.4 unpack_field_and_write_3d_dp()

```
subroutine mo_mpr_restart::unpack_field_and_write::unpack_field_and_write_3d_dp (
            type(ncdataset), intent(inout) nc,
            character(*), intent(in) var_name,
            type(ncdimension), dimension(:), intent(in) var_dims,
            real(dp), intent(in) fill_value,
            real(dp), dimension(:, :, :), intent(in) data,
            logical, dimension(:, :), intent(in) mask,
            character(*), intent(in), optional var_long_name )  [private]
```

The documentation for this interface was generated from the following file:

- mo_mpr_restart.f90

17.121 mo_orderpack::valmed Interface Reference

Public Member Functions

- recursive real(kind=dp) function d_valmed (XDONT)
- recursive real(kind=sp) function r_valmed (XDONT)
- recursive integer(kind=i4) function i_valmed (XDONT)

17.121.1 Member Function/Subroutine Documentation

17.121.1.1 d_valmed()

```
recursive real(kind = dp) function mo_orderpack::valmed::d_valmed (
            real(kind = dp), dimension (:), intent(in) XDONT )
```

17.121.1.2 i_valmed()

```
recursive integer(kind = i4) function mo_orderpack::valmed::i_valmed (
             integer(kind = i4), dimension (:), intent(in) XDONT )
```

17.121.1.3 r_valmed()

```
recursive real(kind = sp) function mo_orderpack::valmed::r_valmed (
             real(kind = sp), dimension (:), intent(in) XDONT )
```

The documentation for this interface was generated from the following file:

- mo_orderpack.f90

17.122 mo_orderpack::valnth Interface Reference

Public Member Functions

- real(kind=dp) function d_valnth (XDONT, NORD)
- real(kind=sp) function r_valnth (XDONT, NORD)
- integer(kind=i4) function i_valnth (XDONT, NORD)

17.122.1 Member Function/Subroutine Documentation

17.122.1.1 d_valnth()

```
real(kind = dp) function mo_orderpack::valnth::d_valnth (
             real(kind = dp), dimension (:), intent(in) XDONT,
             integer(kind = i4), intent(in) NORD )
```

17.122.1.2 i_valnth()

```
integer(kind = i4) function mo_orderpack::valnth::i_valnth (
             integer(kind = i4), dimension (:), intent(in) XDONT,
             integer(kind = i4), intent(in) NORD )
```

17.122.1.3 r_valnth()

```
real(kind = sp) function mo_orderpack::valnth::r_valnth (
             real(kind = sp), dimension (:), intent(in) XDONT,
             integer(kind = i4), intent(in) NORD )
```

The documentation for this interface was generated from the following file:

- mo_orderpack.f90

17.123 mo_ncwrite::var2nc Interface Reference

Extended dump_netcdf for multiple variables.

Public Member Functions

- subroutine var2nc_1d_i4 (f_name, arr, dnames, v_name, dim_unlimited, long_name, units, missing_value, attributes, create, ncid, nrec)
- subroutine var2nc_1d_sp (f_name, arr, dnames, v_name, dim_unlimited, long_name, units, missing_value, attributes, create, ncid, nrec)
- subroutine var2nc_1d_dp (f_name, arr, dnames, v_name, dim_unlimited, long_name, units, missing_value, attributes, create, ncid, nrec)
- subroutine var2nc_2d_i4 (f_name, arr, dnames, v_name, dim_unlimited, long_name, units, missing_value, attributes, create, ncid, nrec)
- subroutine var2nc_2d_sp (f_name, arr, dnames, v_name, dim_unlimited, long_name, units, missing_value, attributes, create, ncid, nrec)
- subroutine var2nc_2d_dp (f_name, arr, dnames, v_name, dim_unlimited, long_name, units, missing_value, attributes, create, ncid, nrec)
- subroutine var2nc_3d_i4 (f_name, arr, dnames, v_name, dim_unlimited, long_name, units, missing_value, attributes, create, ncid, nrec)
- subroutine var2nc_3d_sp (f_name, arr, dnames, v_name, dim_unlimited, long_name, units, missing_value, attributes, create, ncid, nrec)
- subroutine var2nc_3d_dp (f_name, arr, dnames, v_name, dim_unlimited, long_name, units, missing_value, attributes, create, ncid, nrec)
- subroutine var2nc_4d_i4 (f_name, arr, dnames, v_name, dim_unlimited, long_name, units, missing_value, attributes, create, ncid, nrec)
- subroutine var2nc_4d_sp (f_name, arr, dnames, v_name, dim_unlimited, long_name, units, missing_value, attributes, create, ncid, nrec)
- subroutine var2nc_4d_dp (f_name, arr, dnames, v_name, dim_unlimited, long_name, units, missing_value, attributes, create, ncid, nrec)
- subroutine var2nc_5d_i4 (f_name, arr, dnames, v_name, dim_unlimited, long_name, units, missing_value, attributes, create, ncid, nrec)
- subroutine var2nc_5d_sp (f_name, arr, dnames, v_name, dim_unlimited, long_name, units, missing_value, attributes, create, ncid, nrec)
- subroutine var2nc_5d_dp (f_name, arr, dnames, v_name, dim_unlimited, long_name, units, missing_value, attributes, create, ncid, nrec)

17.123.1 Detailed Description

Extended dump_netcdf for multiple variables.

Write different variables including attributes to netcdf file. The attributes are restricted to long_name, units, and missing_value. It is also possible to append variables when an unlimited dimension is specified. call var2nc(f_↩ name, arr, dnames, v_name, dim_unlimited, long_name, units, missing_value, create)

Parameters

in	*character(∗) :: f_name*	filename
in	*integer(i4)/real(sp,dp) :: arr(:[,:[,:[,:[,:]]]])*	array to write
in	*character(∗) :: dnames(:)*	dimension names
in	*character(∗) :: v_name*	variable name
in	*integer(i4), optional :: dim_unlimited*	index of unlimited dimension
in	*character(∗), optional :: long_name*	attribute
in	*character(∗), optional :: units*	attribute

Parameters

in	*integer(i4)/real(sp,dp), optional :: missing_value*	attribute
in	*character(256), dimension(:,:), optional :: attributes*	two dimensional array of attributes size of first dimension equals number of attributes first entry of second dimension equals attribute name (e.g. long_name) second entry of second dimension equals attribute value (e.g. precipitation) every attribute is written as string with the exception of missing_value
in	*logical, optional :: create*	flag - specify whether a output file should be created, default
in,out	*integer(i4)/real(sp,dp), optional :: ncid*	if not given filename will be opened and closed if given and <0 then file will be opened and ncid will return the file unit. if given and >0 then file is assumed open and ncid is used as file unit.
in	*integer(i4), optional :: nrec*	if given: start point on unlimited dimension.

Note

It is not allowed to write the folloing numbers for the indicated type
number | kind
-2.1474836E+09 | integer(i4)
9.9692100E+36 | real(sp)
9.9692099683868690E+36 | real(dp)
These numbers are netcdf fortran 90 constants! They are used to determine the chunksize of the already written variable. Hence, this routine cannot append correctly to variables when these numbers are used. Only five dimensional variables can be written, only one unlimited dimension can be defined.

Author

Stephan Thober & Matthias Cuntz

Date

May 2014

17.123.2 Member Function/Subroutine Documentation

17.123.2.1 var2nc_1d_dp()

```
subroutine mo_ncwrite::var2nc::var2nc_1d_dp (
             character(len = *), intent(in) f_name,
             real(dp), dimension(:), intent(in) arr,
             character(len = *), dimension(:), intent(in) dnames,
             character(len = *), intent(in) v_name,
             integer(i4), intent(in), optional dim_unlimited,
             character(len = *), intent(in), optional long_name,
             character(len = *), intent(in), optional units,
             real(dp), intent(in), optional missing_value,
             character(256), dimension(:, :), intent(in), optional attributes,
             logical, intent(in), optional create,
```

```
            integer(i4), intent(inout), optional ncid,
            integer(i4), intent(in), optional nrec )
```

17.123.2.2 var2nc_1d_i4()

```
subroutine mo_ncwrite::var2nc::var2nc_1d_i4 (
            character(len = *), intent(in) f_name,
            integer(i4), dimension(:), intent(in) arr,
            character(len = *), dimension(:), intent(in) dnames,
            character(len = *), intent(in) v_name,
            integer(i4), intent(in), optional dim_unlimited,
            character(len = *), intent(in), optional long_name,
            character(len = *), intent(in), optional units,
            integer(i4), intent(in), optional missing_value,
            character(256), dimension(:, :), intent(in), optional attributes,
            logical, intent(in), optional create,
            integer(i4), intent(inout), optional ncid,
            integer(i4), intent(in), optional nrec )
```

17.123.2.3 var2nc_1d_sp()

```
subroutine mo_ncwrite::var2nc::var2nc_1d_sp (
            character(len = *), intent(in) f_name,
            real(sp), dimension(:), intent(in) arr,
            character(len = *), dimension(:), intent(in) dnames,
            character(len = *), intent(in) v_name,
            integer(i4), intent(in), optional dim_unlimited,
            character(len = *), intent(in), optional long_name,
            character(len = *), intent(in), optional units,
            real(sp), intent(in), optional missing_value,
            character(256), dimension(:, :), intent(in), optional attributes,
            logical, intent(in), optional create,
            integer(i4), intent(inout), optional ncid,
            integer(i4), intent(in), optional nrec )
```

17.123.2.4 var2nc_2d_dp()

```
subroutine mo_ncwrite::var2nc::var2nc_2d_dp (
            character(len = *), intent(in) f_name,
            real(dp), dimension(:, :), intent(in) arr,
            character(len = *), dimension(:), intent(in) dnames,
            character(len = *), intent(in) v_name,
            integer(i4), intent(in), optional dim_unlimited,
            character(len = *), intent(in), optional long_name,
            character(len = *), intent(in), optional units,
            real(dp), intent(in), optional missing_value,
            character(256), dimension(:, :), intent(in), optional attributes,
            logical, intent(in), optional create,
            integer(i4), intent(inout), optional ncid,
            integer(i4), intent(in), optional nrec )
```

17.123.2.5 var2nc_2d_i4()

```
subroutine mo_ncwrite::var2nc::var2nc_2d_i4 (
             character(len = *), intent(in) f_name,
             integer(i4), dimension(:, :), intent(in) arr,
             character(len = *), dimension(:), intent(in) dnames,
             character(len = *), intent(in) v_name,
             integer(i4), intent(in), optional dim_unlimited,
             character(len = *), intent(in), optional long_name,
             character(len = *), intent(in), optional units,
             integer(i4), intent(in), optional missing_value,
             character(256), dimension(:, :), intent(in), optional attributes,
             logical, intent(in), optional create,
             integer(i4), intent(inout), optional ncid,
             integer(i4), intent(in), optional nrec )
```

17.123.2.6 var2nc_2d_sp()

```
subroutine mo_ncwrite::var2nc::var2nc_2d_sp (
             character(len = *), intent(in) f_name,
             real(sp), dimension(:, :), intent(in) arr,
             character(len = *), dimension(:), intent(in) dnames,
             character(len = *), intent(in) v_name,
             integer(i4), intent(in), optional dim_unlimited,
             character(len = *), intent(in), optional long_name,
             character(len = *), intent(in), optional units,
             real(sp), intent(in), optional missing_value,
             character(256), dimension(:, :), intent(in), optional attributes,
             logical, intent(in), optional create,
             integer(i4), intent(inout), optional ncid,
             integer(i4), intent(in), optional nrec )
```

17.123.2.7 var2nc_3d_dp()

```
subroutine mo_ncwrite::var2nc::var2nc_3d_dp (
             character(len = *), intent(in) f_name,
             real(dp), dimension(:, :, :), intent(in) arr,
             character(len = *), dimension(:), intent(in) dnames,
             character(len = *), intent(in) v_name,
             integer(i4), intent(in), optional dim_unlimited,
             character(len = *), intent(in), optional long_name,
             character(len = *), intent(in), optional units,
             real(dp), intent(in), optional missing_value,
             character(256), dimension(:, :), intent(in), optional attributes,
             logical, intent(in), optional create,
             integer(i4), intent(inout), optional ncid,
             integer(i4), intent(in), optional nrec )
```

17.123.2.8 var2nc_3d_i4()

```
subroutine mo_ncwrite::var2nc::var2nc_3d_i4 (
             character(len = *), intent(in) f_name,
```

```
            integer(i4), dimension(:, :, :), intent(in) arr,
            character(len = *), dimension(:), intent(in) dnames,
            character(len = *), intent(in) v_name,
            integer(i4), intent(in), optional dim_unlimited,
            character(len = *), intent(in), optional long_name,
            character(len = *), intent(in), optional units,
            integer(i4), intent(in), optional missing_value,
            character(256), dimension(:, :), intent(in), optional attributes,
            logical, intent(in), optional create,
            integer(i4), intent(inout), optional ncid,
            integer(i4), intent(in), optional nrec )
```

17.123.2.9 var2nc_3d_sp()

```
subroutine mo_ncwrite::var2nc::var2nc_3d_sp (
            character(len = *), intent(in) f_name,
            real(sp), dimension(:, :, :), intent(in) arr,
            character(len = *), dimension(:), intent(in) dnames,
            character(len = *), intent(in) v_name,
            integer(i4), intent(in), optional dim_unlimited,
            character(len = *), intent(in), optional long_name,
            character(len = *), intent(in), optional units,
            real(sp), intent(in), optional missing_value,
            character(256), dimension(:, :), intent(in), optional attributes,
            logical, intent(in), optional create,
            integer(i4), intent(inout), optional ncid,
            integer(i4), intent(in), optional nrec )
```

17.123.2.10 var2nc_4d_dp()

```
subroutine mo_ncwrite::var2nc::var2nc_4d_dp (
            character(len = *), intent(in) f_name,
            real(dp), dimension(:, :, :, :), intent(in) arr,
            character(len = *), dimension(:), intent(in) dnames,
            character(len = *), intent(in) v_name,
            integer(i4), intent(in), optional dim_unlimited,
            character(len = *), intent(in), optional long_name,
            character(len = *), intent(in), optional units,
            real(dp), intent(in), optional missing_value,
            character(256), dimension(:, :), intent(in), optional attributes,
            logical, intent(in), optional create,
            integer(i4), intent(inout), optional ncid,
            integer(i4), intent(in), optional nrec )
```

17.123.2.11 var2nc_4d_i4()

```
subroutine mo_ncwrite::var2nc::var2nc_4d_i4 (
            character(len = *), intent(in) f_name,
            integer(i4), dimension(:, :, :, :), intent(in) arr,
            character(len = *), dimension(:), intent(in) dnames,
            character(len = *), intent(in) v_name,
            integer(i4), intent(in), optional dim_unlimited,
```

```
            character(len = *), intent(in), optional long_name,
            character(len = *), intent(in), optional units,
            integer(i4), intent(in), optional missing_value,
            character(256), dimension(:, :), intent(in), optional attributes,
            logical, intent(in), optional create,
            integer(i4), intent(inout), optional ncid,
            integer(i4), intent(in), optional nrec )
```

17.123.2.12 var2nc_4d_sp()

```
subroutine mo_ncwrite::var2nc::var2nc_4d_sp (
            character(len = *), intent(in) f_name,
            real(sp), dimension(:, :, :, :), intent(in) arr,
            character(len = *), dimension(:), intent(in) dnames,
            character(len = *), intent(in) v_name,
            integer(i4), intent(in), optional dim_unlimited,
            character(len = *), intent(in), optional long_name,
            character(len = *), intent(in), optional units,
            real(sp), intent(in), optional missing_value,
            character(256), dimension(:, :), intent(in), optional attributes,
            logical, intent(in), optional create,
            integer(i4), intent(inout), optional ncid,
            integer(i4), intent(in), optional nrec )
```

17.123.2.13 var2nc_5d_dp()

```
subroutine mo_ncwrite::var2nc::var2nc_5d_dp (
            character(len = *), intent(in) f_name,
            real(dp), dimension(:, :, :, :, :), intent(in) arr,
            character(len = *), dimension(:), intent(in) dnames,
            character(len = *), intent(in) v_name,
            integer(i4), intent(in), optional dim_unlimited,
            character(len = *), intent(in), optional long_name,
            character(len = *), intent(in), optional units,
            real(dp), intent(in), optional missing_value,
            character(256), dimension(:, :), intent(in), optional attributes,
            logical, intent(in), optional create,
            integer(i4), intent(inout), optional ncid,
            integer(i4), intent(in), optional nrec )
```

17.123.2.14 var2nc_5d_i4()

```
subroutine mo_ncwrite::var2nc::var2nc_5d_i4 (
            character(len = *), intent(in) f_name,
            integer(i4), dimension(:, :, :, :, :), intent(in) arr,
            character(len = *), dimension(:), intent(in) dnames,
            character(len = *), intent(in) v_name,
            integer(i4), intent(in), optional dim_unlimited,
            character(len = *), intent(in), optional long_name,
            character(len = *), intent(in), optional units,
            integer(i4), intent(in), optional missing_value,
            character(256), dimension(:, :), intent(in), optional attributes,
```

```
            logical, intent(in), optional create,
            integer(i4), intent(inout), optional ncid,
            integer(i4), intent(in), optional nrec )
```

17.123.2.15 var2nc_5d_sp()

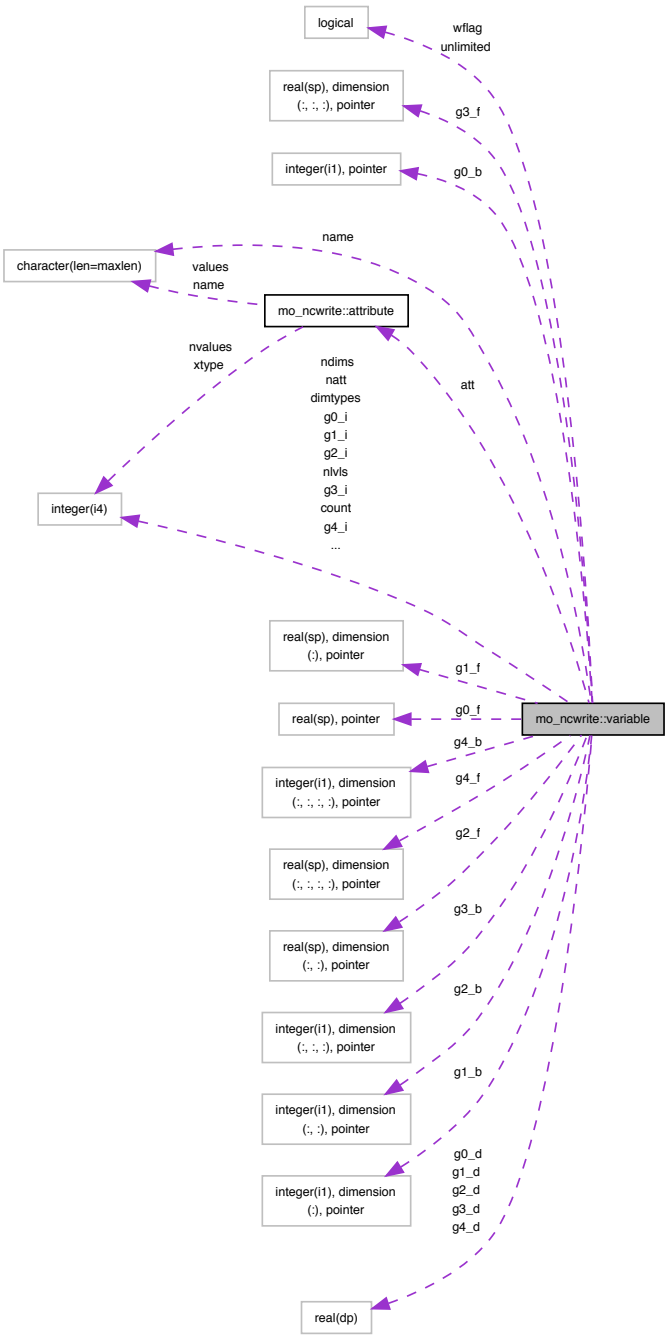
```
subroutine mo_ncwrite::var2nc::var2nc_5d_sp (
            character(len = *), intent(in) f_name,
            real(sp), dimension(:, :, :, :, :), intent(in) arr,
            character(len = *), dimension(:), intent(in) dnames,
            character(len = *), intent(in) v_name,
            integer(i4), intent(in), optional dim_unlimited,
            character(len = *), intent(in), optional long_name,
            character(len = *), intent(in), optional units,
            real(sp), intent(in), optional missing_value,
            character(256), dimension(:, :), intent(in), optional attributes,
            logical, intent(in), optional create,
            integer(i4), intent(inout), optional ncid,
            integer(i4), intent(in), optional nrec )
```

The documentation for this interface was generated from the following file:

- mo_ncwrite.f90

17.124 mo_ncwrite::variable Type Reference

Collaboration diagram for mo_ncwrite::variable:



Public Attributes

- character(len=maxlen) name
- integer(i4) xtype
- integer(i4) nlvls

- integer(i4) nsubs
- logical unlimited
- integer(i4) varid
- integer(i4) ndims
- integer(i4), dimension(nmaxdim) dimids
- integer(i4), dimension(nmaxdim) dimtypes
- integer(i4) natt
- type(attribute), dimension(nmaxatt) att
- integer(i4), dimension(nmaxdim) start
- integer(i4), dimension(nmaxdim) count
- logical wflag
- integer(i1), pointer g0_b
- integer(i1), dimension(:), pointer g1_b
- integer(i1), dimension(:, :), pointer g2_b
- integer(i1), dimension(:, :, :), pointer g3_b
- integer(i1), dimension(:, :, :, :), pointer g4_b
- integer(i4), pointer g0_i
- integer(i4), dimension(:), pointer g1_i
- integer(i4), dimension(:, :), pointer g2_i
- integer(i4), dimension(:, :, :), pointer g3_i
- integer(i4), dimension(:, :, :, :), pointer g4_i
- real(sp), pointer g0_f
- real(sp), dimension(:), pointer g1_f
- real(sp), dimension(:, :), pointer g2_f
- real(sp), dimension(:, :, :), pointer g3_f
- real(sp), dimension(:, :, :, :), pointer g4_f
- real(dp), pointer g0_d
- real(dp), dimension(:), pointer g1_d
- real(dp), dimension(:, :), pointer g2_d
- real(dp), dimension(:, :, :), pointer g3_d
- real(dp), dimension(:, :, :, :), pointer g4_d

17.124.1 Member Data Documentation

17.124.1.1 att

```
type(attribute), dimension(nmaxatt) mo_ncwrite::variable::att
```

17.124.1.2 count

```
integer(i4), dimension(nmaxdim) mo_ncwrite::variable::count
```

17.124.1.3 dimids

```
integer(i4), dimension(nmaxdim) mo_ncwrite::variable::dimids
```

17.124.1.4 dimtypes

```
integer(i4), dimension(nmaxdim) mo_ncwrite::variable::dimtypes
```

17.124.1.5 g0_b

```
integer(i1), pointer mo_ncwrite::variable::g0_b
```

17.124.1.6 g0_d

```
real(dp), pointer mo_ncwrite::variable::g0_d
```

17.124.1.7 g0_f

```
real(sp), pointer mo_ncwrite::variable::g0_f
```

17.124.1.8 g0_i

```
integer(i4), pointer mo_ncwrite::variable::g0_i
```

17.124.1.9 g1_b

```
integer(i1), dimension(:), pointer mo_ncwrite::variable::g1_b
```

17.124.1.10 g1_d

```
real(dp), dimension(:), pointer mo_ncwrite::variable::g1_d
```

17.124.1.11 g1_f

```
real(sp), dimension(:), pointer mo_ncwrite::variable::g1_f
```

17.124.1.12 g1_i

```
integer(i4), dimension(:), pointer mo_ncwrite::variable::g1_i
```

17.124.1.13 g2_b

```
integer(i1), dimension(:, :), pointer mo_ncwrite::variable::g2_b
```

17.124.1.14 g2_d

```
real(dp), dimension(:, :), pointer mo_ncwrite::variable::g2_d
```

17.124.1.15 g2_f

```
real(sp), dimension(:, :), pointer mo_ncwrite::variable::g2_f
```

17.124.1.16 g2_i

```
integer(i4), dimension(:, :), pointer mo_ncwrite::variable::g2_i
```

17.124.1.17 g3_b

```
integer(i1), dimension(:, :, :), pointer mo_ncwrite::variable::g3_b
```

17.124.1.18 g3_d

```
real(dp), dimension(:, :, :), pointer mo_ncwrite::variable::g3_d
```

17.124.1.19 g3_f

```
real(sp), dimension(:, :, :), pointer mo_ncwrite::variable::g3_f
```

17.124.1.20 g3_i

```
integer(i4), dimension(:, :, :), pointer mo_ncwrite::variable::g3_i
```

17.124.1.21 g4_b

```
integer(i1), dimension(:, :, :, :), pointer mo_ncwrite::variable::g4_b
```

17.124.1.22 g4_d

```
real(dp), dimension(:, :, :, :), pointer mo_ncwrite::variable::g4_d
```

17.124.1.23 g4_f

```
real(sp), dimension(:, :, :, :), pointer mo_ncwrite::variable::g4_f
```

17.124.1.24 g4_i

```
integer(i4), dimension(:, :, :, :), pointer mo_ncwrite::variable::g4_i
```

17.124.1.25 name

```
character (len = maxlen) mo_ncwrite::variable::name
```

17.124.1.26 natt

```
integer(i4) mo_ncwrite::variable::natt
```

17.124.1.27 ndims

```
integer(i4) mo_ncwrite::variable::ndims
```

17.124.1.28 nlvls

```
integer(i4) mo_ncwrite::variable::nlvls
```

17.124.1.29 nsubs

```
integer(i4) mo_ncwrite::variable::nsubs
```

17.124.1.30 start

```
integer(i4), dimension(nmaxdim) mo_ncwrite::variable::start
```

17.124.1.31 unlimited

```
logical mo_ncwrite::variable::unlimited
```

17.124.1.32 varid

```
integer(i4) mo_ncwrite::variable::varid
```

17.124.1.33 wflag

```
logical mo_ncwrite::variable::wflag
```

17.124.1.34 xtype

```
integer(i4) mo_ncwrite::variable::xtype
```

The documentation for this type was generated from the following file:

- mo_ncwrite.f90

17.125 mo_moment::variance Interface Reference

Public Member Functions

- real(sp) function variance_sp (dat, mask)
- real(dp) function variance_dp (dat, mask)

17.125.1 Member Function/Subroutine Documentation

17.125.1.1 variance_dp()

```
real(dp) function mo_moment::variance::variance_dp (
             real(dp), dimension(:), intent(in) dat,
             logical, dimension(:), intent(in), optional mask )
```

17.125.1.2 variance_sp()

```
real(sp) function mo_moment::variance::variance_sp (
             real(sp), dimension(:), intent(in) dat,
             logical, dimension(:), intent(in), optional mask )
```

The documentation for this interface was generated from the following file:

- mo_moment.f90

17.126 mo_errormeasures::wnse Interface Reference

Public Member Functions

- real(sp) function wnse_sp_1d (x, y, mask)
- real(dp) function wnse_dp_1d (x, y, mask)
- real(dp) function wnse_dp_2d (x, y, mask)
- real(sp) function wnse_sp_2d (x, y, mask)
- real(sp) function wnse_sp_3d (x, y, mask)
- real(dp) function wnse_dp_3d (x, y, mask)

17.126.1 Member Function/Subroutine Documentation

17.126.1.1 wnse_dp_1d()

```
real(dp) function mo_errormeasures::wnse::wnse_dp_1d (
            real(dp), dimension(:), intent(in) x,
            real(dp), dimension(:), intent(in) y,
            logical, dimension(:), intent(in), optional mask )
```

17.126.1.2 wnse_dp_2d()

```
real(dp) function mo_errormeasures::wnse::wnse_dp_2d (
            real(dp), dimension(:,:), intent(in) x,
            real(dp), dimension(:,:), intent(in) y,
            logical, dimension(:,:), intent(in), optional mask )
```

17.126.1.3 wnse_dp_3d()

```
real(dp) function mo_errormeasures::wnse::wnse_dp_3d (
            real(dp), dimension(:,:,:), intent(in) x,
            real(dp), dimension(:,:,:), intent(in) y,
            logical, dimension(:,:,:), intent(in), optional mask )
```

17.126.1.4 wnse_sp_1d()

```
real(sp) function mo_errormeasures::wnse::wnse_sp_1d (
            real(sp), dimension(:), intent(in) x,
            real(sp), dimension(:), intent(in) y,
            logical, dimension(:), intent(in), optional mask )
```

17.126.1.5 wnse_sp_2d()

```
real(sp) function mo_errormeasures::wnse::wnse_sp_2d (
            real(sp), dimension(:,:), intent(in) x,
```

```
            real(sp), dimension(:,:), intent(in) y,
            logical, dimension(:,:), intent(in), optional mask )
```

17.126.1.6 wnse_sp_3d()

```
real(sp) function mo_errormeasures::wnse::wnse_sp_3d (
            real(sp), dimension(:,:,:), intent(in) x,
            real(sp), dimension(:,:,:), intent(in) y,
            logical, dimension(:,:,:), intent(in), optional mask )
```

The documentation for this interface was generated from the following file:

- mo_errormeasures.f90

17.127 mo_xor4096::xor4096 Interface Reference

Public Member Functions

- subroutine xor4096s_0d (seed, SingleIntegerRN, save_state)
- subroutine xor4096s_1d (seed, SingleIntegerRN, save_state)
- subroutine xor4096f_0d (seed, SingleRealRN, save_state)
- subroutine xor4096f_1d (seed, SingleRealRN, save_state)
- subroutine xor4096l_0d (seed, DoubleIntegerRN, save_state)
- subroutine xor4096l_1d (seed, DoubleIntegerRN, save_state)
- subroutine xor4096d_0d (seed, DoubleRealRN, save_state)
- subroutine xor4096d_1d (seed, DoubleRealRN, save_state)

17.127.1 Member Function/Subroutine Documentation

17.127.1.1 xor4096d_0d()

```
subroutine mo_xor4096::xor4096::xor4096d_0d (
            integer(i8), intent(in) seed,
            real(dp), intent(out) DoubleRealRN,
            integer(i8), dimension(n_save_state), intent(inout), optional save_state )
```

17.127.1.2 xor4096d_1d()

```
subroutine mo_xor4096::xor4096::xor4096d_1d (
            integer(i8), dimension(:), intent(in) seed,
            real(dp), dimension(size(seed, 1)), intent(out) DoubleRealRN,
            integer(i8), dimension(size(seed, 1), n_save_state), intent(inout), optional
save_state )
```

17.127.1.3 xor4096f_0d()

```
subroutine mo_xor4096::xor4096::xor4096f_0d (
             integer(i4), intent(in) seed,
             real(sp), intent(out) SingleRealRN,
             integer(i4), dimension(n_save_state), intent(inout), optional save_state )
```

17.127.1.4 xor4096f_1d()

```
subroutine mo_xor4096::xor4096::xor4096f_1d (
             integer(i4), dimension(:), intent(in) seed,
             real(sp), dimension(size(seed)), intent(out) SingleRealRN,
             integer(i4), dimension(size(seed, 1), n_save_state), intent(inout), optional
save_state )
```

17.127.1.5 xor4096l_0d()

```
subroutine mo_xor4096::xor4096::xor4096l_0d (
             integer(i8), intent(in) seed,
             integer(i8), intent(out) DoubleIntegerRN,
             integer(i8), dimension(n_save_state), intent(inout), optional save_state )
```

17.127.1.6 xor4096l_1d()

```
subroutine mo_xor4096::xor4096::xor4096l_1d (
             integer(i8), dimension(:), intent(in) seed,
             integer(i8), dimension(size(seed, 1)), intent(out) DoubleIntegerRN,
             integer(i8), dimension(size(seed, 1), n_save_state), intent(inout), optional
save_state )
```

17.127.1.7 xor4096s_0d()

```
subroutine mo_xor4096::xor4096::xor4096s_0d (
             integer(i4), intent(in) seed,
             integer(i4), intent(out) SingleIntegerRN,
             integer(i4), dimension(n_save_state), intent(inout), optional save_state )
```

17.127.1.8 xor4096s_1d()

```
subroutine mo_xor4096::xor4096::xor4096s_1d (
             integer(i4), dimension(:), intent(in) seed,
             integer(i4), dimension(size(seed, 1)), intent(out) SingleIntegerRN,
             integer(i4), dimension(size(seed, 1), n_save_state), intent(inout), optional
save_state )
```

The documentation for this interface was generated from the following file:

- mo_xor4096.f90

17.128 mo_xor4096::xor4096g Interface Reference

Public Member Functions

- subroutine xor4096gf_0d (seed, SingleRealRN, save_state)
- subroutine xor4096gf_1d (seed, SingleRealRN, save_state)
- subroutine xor4096gd_0d (seed, DoubleRealRN, save_state)
- subroutine xor4096gd_1d (seed, DoubleRealRN, save_state)

17.128.1 Member Function/Subroutine Documentation

17.128.1.1 xor4096gd_0d()

```
subroutine mo_xor4096::xor4096g::xor4096gd_0d (
             integer(i8), intent(in) seed,
             real(dp), intent(out) DoubleRealRN,
             integer(i8), dimension(n_save_state), intent(inout), optional save_state )
```

17.128.1.2 xor4096gd_1d()

```
subroutine mo_xor4096::xor4096g::xor4096gd_1d (
             integer(i8), dimension(:), intent(in) seed,
             real(dp), dimension(size(seed)), intent(out) DoubleRealRN,
             integer(i8), dimension(size(seed), n_save_state), intent(inout), optional save_↩
state )
```

17.128.1.3 xor4096gf_0d()

```
subroutine mo_xor4096::xor4096g::xor4096gf_0d (
             integer(i4), intent(in) seed,
             real(sp), intent(out) SingleRealRN,
             integer(i4), dimension(n_save_state), intent(inout), optional save_state )
```

17.128.1.4 xor4096gf_1d()

```
subroutine mo_xor4096::xor4096g::xor4096gf_1d (
             integer(i4), dimension(:), intent(in) seed,
             real(sp), dimension(size(seed)), intent(out) SingleRealRN,
             integer(i4), dimension(size(seed), n_save_state), intent(inout), optional save_↩
state )
```

The documentation for this interface was generated from the following file:

- mo_xor4096.f90

Chapter 18

File Documentation

18.1 1-main.dox File Reference

18.2 2-get_started.dox File Reference

18.3 3-data_preparation.dox File Reference

18.4 4-visualise_out.dox File Reference

18.5 5-calibration.dox File Reference

18.6 6-style_guide.dox File Reference

18.7 7-test_basin.dox File Reference

18.8 8-protocols_for_setup_new_mHM_basin.dox File Reference

18.9 9-mRM.dox File Reference

18.10 DEPENDENCIES.md File Reference

18.11 mhm_driver.f90 File Reference

Functions/Subroutines

- program mhm_driver

  Distributed precipitation-runoff model mHM.

18.11.1 Function/Subroutine Documentation

18.11.1.1 mhm_driver()

```
program mhm_driver ( )
```

Distributed precipitation-runoff model mHM.

This is the main driver of mHM, which calls one instance of mHM for a multiple basins and a given period.

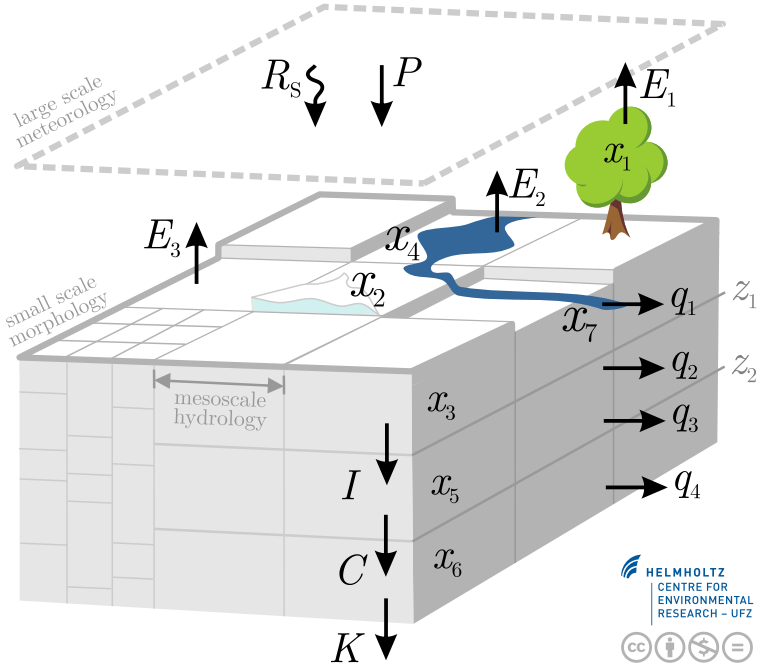


Figure 18.1: Typical mHM cell

Luis Samaniego & Rohini Kumar (UFZ)

Date

Jun 2018

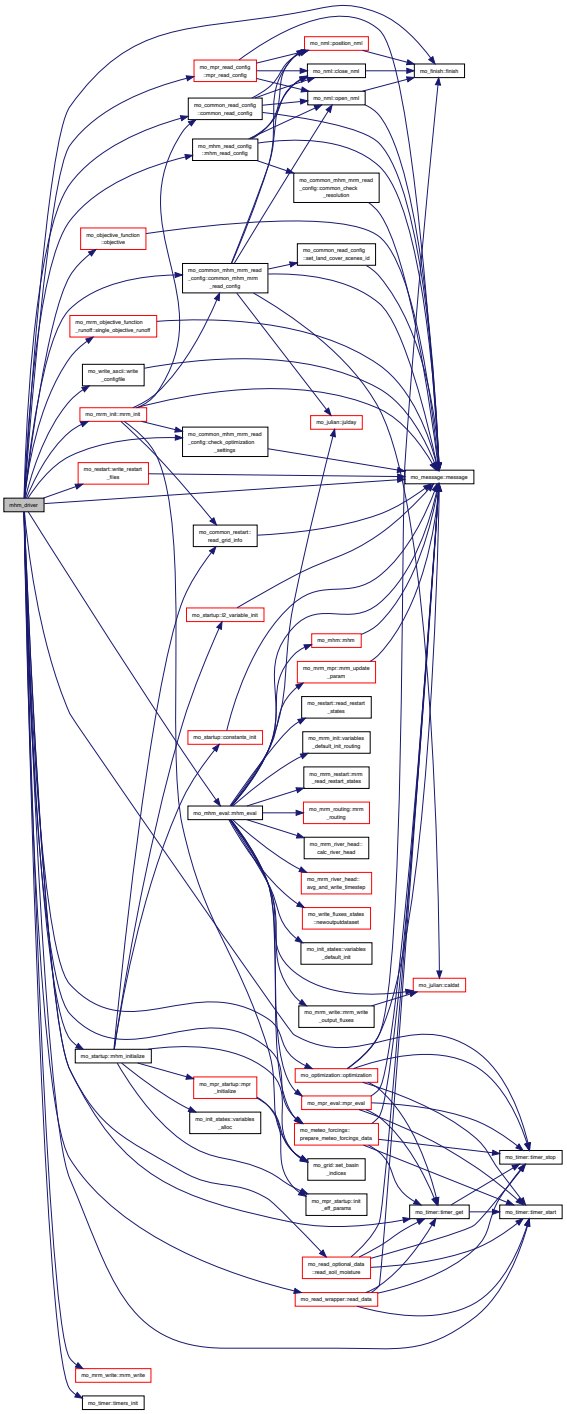
Version

5.9

Copyright

References mo_common_mhm_mrm_read_config::check_optimization_settings(), mo_common_mhm_mrm_↩
read_config::common_mhm_mrm_read_config(), mo_common_read_config::common_read_config(), mo_global↩
_variables::dirprecipitation, mo_kind::dp, mo_file::file_main, mo_finish::finish(), mo_kind::i4, mo_message↩
::message(), mo_message::message_text, mo_mhm_eval::mhm_eval(), mo_startup::mhm_initialize(), mo_mhm↩
_read_config::mhm_read_config(), mo_mpr_read_config::mpr_read_config(), mo_mrm_init::mrm_init(), mo_mrm↩
_write::mrm_write(), mo_common_mhm_mrm_variables::ntstepday, mo_objective_function::objective(), mo_↩
optimization::optimization(), mo_meteo_forcings::prepare_meteo_forcings_data(), mo_read_wrapper::read_data(),
mo_read_optional_data::read_soil_moisture(), mo_string_utils::separator, mo_mrm_objective_function_runoff↩
::single_objective_runoff(), mo_timer::timer_get(), mo_timer::timer_start(), mo_timer::timer_stop(), mo_timer↩
::timers_init(), mo_file::version, mo_file::version_date, mo_write_ascii::write_configfile(), mo_common_variables↩
::write_restart, and mo_restart::write_restart_files().

Here is the call graph for this function:



18.12 mhm_papers.md File Reference

18.13 mo_anneal.f90 File Reference

Data Types

- interface mo_anneal::anneal

  anneal
- interface mo_anneal::gettemperature

  GetTemperature.
- interface mo_anneal::generate_neighborhood_weight

Modules

- module mo_anneal

Functions/Subroutines

- real(dp) function, dimension(size(para, 1)) mo_anneal::anneal_dp (eval, cost, para, prange, prange_func, temp, Dt, nITERmax, Len, nST, eps, acc, seeds, printflag, maskpara, weight, changeParaMode, reflection↩Flag, pertubFlexFlag, maxit, undef_funcval, tmp_file, funcbest, history)
- real(dp) function mo_anneal::gettemperature_dp (paraset, cost, eval, acc_goal, prange, prange_func, samplesize, maskpara, seeds, printflag, weight, maxit, undef_funcval)
- real(dp) function mo_anneal::pargen_anneal_dp (old, dMax, oMin, oMax, RN)
- real(dp) function mo_anneal::pargen_dds_dp (old, perturb, oMin, oMax, RN)
- real(dp) function mo_anneal::dchange_dp (delta, iDigit, isZero)
- subroutine mo_anneal::generate_neighborhood_weight_dp (truepara, cum_weight, save_state_xor, iTotal↩Counter, nITERmax, neighborhood)

18.14 mo_append.f90 File Reference

Data Types

- interface mo_append::append

  Append (rows) scalars, vectors, and matrixes onto existing array.
- interface mo_append::paste

  Paste (columns) scalars, vectors, and matrixes onto existing array.

Modules

- module mo_append

  Append values on existing arrays.

Functions/Subroutines

- subroutine mo_append::append_i4_v_s (vec1, sca2)
- subroutine mo_append::append_i4_v_v (vec1, vec2)
- subroutine mo_append::append_i4_m_m (mat1, mat2, fill_value)
- subroutine mo_append::append_i4_3d (mat1, mat2, fill_value)
- subroutine mo_append::append_i8_v_s (vec1, sca2)
- subroutine mo_append::append_i8_v_v (vec1, vec2)
- subroutine mo_append::append_i8_m_m (mat1, mat2, fill_value)
- subroutine mo_append::append_i8_3d (mat1, mat2, fill_value)
- subroutine mo_append::append_sp_v_s (vec1, sca2)
- subroutine mo_append::append_sp_v_v (vec1, vec2)

- subroutine mo_append::append_sp_m_m (mat1, mat2, fill_value)
- subroutine mo_append::append_sp_3d (mat1, mat2, fill_value)
- subroutine mo_append::append_dp_v_s (vec1, sca2)
- subroutine mo_append::append_dp_v_v (vec1, vec2)
- subroutine mo_append::append_dp_m_m (mat1, mat2, fill_value)
- subroutine mo_append::append_dp_3d (mat1, mat2, fill_value)
- subroutine mo_append::append_char_v_s (vec1, sca2)
- subroutine mo_append::append_char_v_v (vec1, vec2)
- subroutine mo_append::append_char_m_m (mat1, mat2, fill_value)
- subroutine mo_append::append_char_3d (mat1, mat2, fill_value)
- subroutine mo_append::append_lgt_v_s (vec1, sca2)
- subroutine mo_append::append_lgt_v_v (vec1, vec2)
- subroutine mo_append::append_lgt_m_m (mat1, mat2, fill_value)
- subroutine mo_append::append_lgt_3d (mat1, mat2, fill_value)
- subroutine mo_append::paste_i4_m_s (mat1, sca2)
- subroutine mo_append::paste_i4_m_v (mat1, vec2, fill_value)
- subroutine mo_append::paste_i4_m_m (mat1, mat2, fill_value)
- subroutine mo_append::paste_i8_m_s (mat1, sca2)
- subroutine mo_append::paste_i8_m_v (mat1, vec2, fill_value)
- subroutine mo_append::paste_i8_m_m (mat1, mat2, fill_value)
- subroutine mo_append::paste_sp_m_s (mat1, sca2)
- subroutine mo_append::paste_sp_m_v (mat1, vec2, fill_value)
- subroutine mo_append::paste_sp_m_m (mat1, mat2, fill_value)
- subroutine mo_append::paste_dp_m_s (mat1, sca2)
- subroutine mo_append::paste_dp_m_v (mat1, vec2, fill_value)
- subroutine mo_append::paste_dp_m_m (mat1, mat2, fill_value)
- subroutine mo_append::paste_char_m_s (mat1, sca2)
- subroutine mo_append::paste_char_m_v (mat1, vec2, fill_value)
- subroutine mo_append::paste_char_m_m (mat1, mat2, fill_value)
- subroutine mo_append::paste_lgt_m_s (mat1, sca2)
- subroutine mo_append::paste_lgt_m_v (mat1, vec2)
- subroutine mo_append::paste_lgt_m_m (mat1, mat2)

18.15 mo_canopy_interc.f90 File Reference

Modules

- module mo_canopy_interc

  Canopy interception.

Functions/Subroutines

- elemental pure subroutine, public mo_canopy_interc::canopy_interc (pet, interc_max, precip, interc, throughfall, evap_canopy)

  Canopy interception.

18.16 mo_common_constants.f90 File Reference

Modules

- module mo_common_constants

  Provides constants commonly used by mHM, mRM and MPR.

Variables

- real(dp), parameter, public mo_common_constants::eps_dp = epsilon(1.0_dp)

  epsilon(1.0) in double precision
- real(sp), parameter, public mo_common_constants::eps_sp = epsilon(1.0_sp)

  epsilon(1.0) in single precision
- integer(i4), parameter, public mo_common_constants::nodata_i4 = -9999_i4
- real(dp), parameter, public mo_common_constants::nodata_dp = -9999._dp
- real(dp), parameter, public mo_common_constants::p1_initstatefluxes = 0.00_dp
- integer(i4), parameter, public mo_common_constants::ncolpars = 5_i4
- integer(i4), parameter, public mo_common_constants::maxnobasins = 50_i4
- integer(i4), parameter, public mo_common_constants::maxnlcovers = 50_i4
- real(dp), parameter, public mo_common_constants::dayhours = 24.0_dp
- real(dp), parameter, public mo_common_constants::yearmonths = 12.0_dp
- integer(i4), parameter, public mo_common_constants::yearmonths_i4 = 12
- real(dp), parameter, public mo_common_constants::yeardays = 365.0_dp
- real(dp), parameter, public mo_common_constants::daysecs = 86400.0_dp
- real(dp), parameter, public mo_common_constants::hoursecs = 3600.0_dp

18.17 mo_common_file.f90 File Reference

Modules

- module mo_common_file

  Provides file names and units for mRM.

Variables

- character(len= *), parameter mo_common_file::file_dem = 'dem.asc'

  DEM input data file.
- integer, parameter mo_common_file::udem = 53

  Unit for DEM input data file.
- integer, parameter mo_common_file::ulcoverclass = 61

  Unit for LCover input data file.
- character(len= *), parameter mo_common_file::file_config = 'ConfigFile.log'

  file defining mHM's outputs
- integer, parameter mo_common_file::uconfig = 68

  Unit for file defining mHM's outputs.

18.18 mo_common_functions.f90 File Reference

Modules

- module mo_common_functions

  Provides small utility functions used by multiple parts of the code (mHM, mRM, MPR)

Functions/Subroutines

- logical function, public mo_common_functions::in_bound (params)

  TODO: add description.

18.19 mo_common_mHM_mRM_file.f90 File Reference

Modules

- module mo_common_mhm_mrm_file

  Provides file names and units for mHM.

Variables

- character(len=∗), parameter mo_common_mhm_mrm_file::file_opti = 'FinalParam.out'

  file defining optimization outputs (objective and parameter set)
- integer, parameter mo_common_mhm_mrm_file::uopti = 72

  Unit for file optimization outputs (objective and parameter set)
- character(len=∗), parameter mo_common_mhm_mrm_file::file_opti_nml = 'FinalParam.nml'

  file defining optimization outputs in a namelist format (parameter set)
- integer, parameter mo_common_mhm_mrm_file::uopti_nml = 73

  Unit for file optimization outputs in a namelist format (parameter set)

18.20 mo_common_mHM_mRM_read_config.f90 File Reference

Modules

- module mo_common_mhm_mrm_read_config

  Reading of main model configurations.

Functions/Subroutines

- subroutine, public mo_common_mhm_mrm_read_config::common_mhm_mrm_read_config (file_namelist, unamelist)

  Read main configurations for common parts.
- subroutine, public mo_common_mhm_mrm_read_config::check_optimization_settings

  TODO: add description.
- subroutine, public mo_common_mhm_mrm_read_config::common_check_resolution (do_message, allow↩ _subgrid_routing)

  TODO: add description.

18.21 mo_common_mHM_mRM_variables.f90 File Reference

Modules

- module mo_common_mhm_mrm_variables

  Provides structures needed by mHM, mRM and/or mpr.

Variables

- integer(i4) mo_common_mhm_mrm_variables::mrm_coupling_mode
- integer(i4), public mo_common_mhm_mrm_variables::timestep
- real(dp), public mo_common_mhm_mrm_variables::c2tstu
- real(dp), dimension(:), allocatable, public mo_common_mhm_mrm_variables::resolutionrouting

- logical, public mo_common_mhm_mrm_variables::read_restart
- type(period), dimension(:), allocatable, public mo_common_mhm_mrm_variables::warmper
- type(period), dimension(:), allocatable, public mo_common_mhm_mrm_variables::evalper
- type(period), dimension(:), allocatable, public mo_common_mhm_mrm_variables::simper
- type(period), public mo_common_mhm_mrm_variables::readper
- integer(i4), dimension(:), allocatable, public mo_common_mhm_mrm_variables::warmingdays
- integer(i4), dimension(:, :), allocatable, public mo_common_mhm_mrm_variables::lcyearid
- integer(i4), public mo_common_mhm_mrm_variables::ntstepday
- character(256), dimension(:), allocatable, public mo_common_mhm_mrm_variables::dirrestartin
- integer(i4), public mo_common_mhm_mrm_variables::opti_method
- integer(i4), public mo_common_mhm_mrm_variables::opti_function
- logical, public mo_common_mhm_mrm_variables::optimize
- logical, public mo_common_mhm_mrm_variables::optimize_restart
- integer(i8), public mo_common_mhm_mrm_variables::seed
- integer(i4), public mo_common_mhm_mrm_variables::niterations
- real(dp), public mo_common_mhm_mrm_variables::dds_r
- real(dp), public mo_common_mhm_mrm_variables::sa_temp
- integer(i4), public mo_common_mhm_mrm_variables::sce_ngs
- integer(i4), public mo_common_mhm_mrm_variables::sce_npg
- integer(i4), public mo_common_mhm_mrm_variables::sce_nps
- logical, public mo_common_mhm_mrm_variables::mcmc_opti
- integer(i4), parameter, public mo_common_mhm_mrm_variables::nerror_model = 2
- real(dp), dimension(nerror_model), public mo_common_mhm_mrm_variables::mcmc_error_params

18.22 mo_common_read_config.f90 File Reference

Modules

- module mo_common_read_config

  Reading of main model configurations.

Functions/Subroutines

- subroutine, public mo_common_read_config::common_read_config (file_namelist, unamelist)

  Read main configurations commonly used by mHM, mRM and MPR.
- subroutine, public mo_common_read_config::set_land_cover_scenes_id (sim_Per, LCyear_Id, LCfilename)

  Read main configurations commonly used by mHM, mRM and MPR.

18.23 mo_common_read_data.f90 File Reference

Modules

- module mo_common_read_data

  TODO: add description.

Functions/Subroutines

- subroutine, public mo_common_read_data::read_dem

  TODO: add description.
- subroutine, public mo_common_read_data::read_lcover

  TODO: add description.

18.24 mo_common_restart.f90 File Reference

Modules

- module mo_common_restart

  TODO: add description.

Functions/Subroutines

- subroutine, public mo_common_restart::write_grid_info (grid_in, level_name, nc)

  write restart files for each basin
- subroutine, public mo_common_restart::read_grid_info (iBasin, InPath, level_name, fname_part, new_grid)

  reads configuration apart from Level 11 configuration from a restart directory

18.25 mo_common_variables.f90 File Reference

Data Types

- type mo_common_variables::period
- type mo_common_variables::grid
- type mo_common_variables::gridremapper

Modules

- module mo_common_variables

  Provides structures needed by mHM, mRM and/or mpr.

Variables

- character(1024), public mo_common_variables::project_details
- character(1024), public mo_common_variables::setup_description
- character(1024), public mo_common_variables::simulation_type
- character(256), public mo_common_variables::conventions
- character(1024), public mo_common_variables::contact
- character(1024), public mo_common_variables::mhm_details
- character(1024), public mo_common_variables::history
- integer(i4), public mo_common_variables::iflag_cordinate_sys
- real(dp), dimension(:), allocatable, public mo_common_variables::resolutionhydrology
- integer(i4), dimension(:), allocatable, public mo_common_variables::l0_basin
- integer(i4), dimension(:), allocatable, public mo_common_variables::l11_basin
- logical, public mo_common_variables::write_restart
- character(256), dimension(:), allocatable, public mo_common_variables::dirrestartout
- character(256), public mo_common_variables::dirconfigout
- character(256), public mo_common_variables::dircommonfiles
- character(256), dimension(:), allocatable, public mo_common_variables::dirmorpho
- character(256), dimension(:), allocatable, public mo_common_variables::dirlcover
- character(256), dimension(:), allocatable, public mo_common_variables::dirout
- character(256), dimension(:), allocatable, public mo_common_variables::filelatlon
- type(grid), dimension(:), allocatable, target, public mo_common_variables::level0
- type(grid), dimension(:), allocatable, target, public mo_common_variables::level1
- type(gridremapper), dimension(:), allocatable, public mo_common_variables::l0_l1_remap

- type(gridremapper), dimension(:), allocatable, public mo_common_variables::l0_l11_remap
- type(gridremapper), dimension(:), allocatable, public mo_common_variables::l1_l11_remap
- real(dp), dimension(:), allocatable, public mo_common_variables::l0_elev
- integer(i4), dimension(:, :), allocatable, public mo_common_variables::l0_lcover
- integer(i4), public mo_common_variables::nbasins
- integer(i4), public mo_common_variables::nuniquel0basins
- integer(i4), public mo_common_variables::nlcoverscene
- character(256), dimension(:), allocatable, public mo_common_variables::lcfilename
- integer(i4), dimension(:), allocatable, public mo_common_variables::lc_year_start
- integer(i4), dimension(:), allocatable, public mo_common_variables::lc_year_end
- integer(i4), parameter, public mo_common_variables::nprocesses = 10
- integer(i4), dimension(nprocesses, 3), public mo_common_variables::processmatrix
- real(dp), dimension(:, :), allocatable, target, public mo_common_variables::global_parameters
- character(256), dimension(:), allocatable, public mo_common_variables::global_parameters_name
- logical mo_common_variables::alma_convention

18.26 mo_constants.f90 File Reference

Modules

- module mo_constants

  Provides computational, mathematical, physical, and file constants.

Variables

- real(dp), parameter mo_constants::pi_dp = 3.14159265358979323846264338327950288419 7_dp

  Pi in double precision.
- real(sp), parameter mo_constants::pi_sp = 3.14159265358979323846264338327950288419 7_sp

  Pi in single precision.
- real(dp), parameter mo_constants::pio2_dp = 1.57079632679489661923132169163975144209858_dp

  Pi/2 in double precision.
- real(sp), parameter mo_constants::pio2_sp = 1.57079632679489661923132169163975144209858_sp

  Pi/2 in single precision.
- real(dp), parameter mo_constants::twopi_dp = 6.28318530717958647692528676655900576839 4_dp

  2∗Pi in double precision
- real(sp), parameter mo_constants::twopi_sp = 6.28318530717958647692528676655900576839 4_sp

  2∗Pi in single precision
- real(dp), parameter mo_constants::sqrt2_dp = 1.41421356237309504880168872420969807856967_dp

  Square root of 2 in double precision.
- real(sp), parameter mo_constants::sqrt2_sp = 1.41421356237309504880168872420969807856967_sp

  Square root of 2 in single precision.
- real(dp), parameter mo_constants::twothird_dp = 0.66666666666666666666666666666666666667_dp

  2/3 in double precision
- real(sp), parameter mo_constants::twothird_sp = 0.66666666666666666666666666666666666667_sp

  2/3 in single precision
- real(dp), parameter mo_constants::deg2rad_dp = PI_dp / 180._dp

  degree to radian conversion (pi/180) in double precision
- real(sp), parameter mo_constants::deg2rad_sp = PI_sp / 180._sp

  degree to radian conversion (pi/180) in double precision
- real(dp), parameter mo_constants::rad2deg_dp = 180._dp / PI_dp

  radian to conversion (180/pi) in double precision

- real(sp), parameter mo_constants::rad2deg_sp = 180._sp / PI_sp

  radian to degree conversion (180/pi) in single precision
- real(sp), parameter, public mo_constants::secday_sp = 86400.0_sp

  Time conversion Seconds per day [s] in single precision.
- real(dp), parameter, public mo_constants::secday_dp = 86400.0_dp
- real(dp), parameter, public mo_constants::dayhours = 24.0_dp
- real(dp), parameter, public mo_constants::yearmonths = 12.0_dp
- real(dp), parameter, public mo_constants::yeardays = 365.0_dp
- real(dp), parameter, public mo_constants::daysecs = 86400.0_dp
- real(dp), parameter mo_constants::psychro_dp = 0.0646_dp

  Psychrometric constant [kPa K^-1] in double precision.
- real(sp), parameter mo_constants::psychro_sp = 0.0646_sp

  Psychrometric constant [kPa K^-1] in sibgle precision.
- real(dp), parameter mo_constants::gravity_dp = 9.81_dp

  Gravity accelaration [m^2 s^-1] in double precision.
- real(sp), parameter mo_constants::gravity_sp = 9.81_sp

  Gravity accelaration [m^2 s^-1] in single precision.
- real(dp), parameter mo_constants::solarconst_dp = 1367._dp

  Solar constant in [J m^-2 s^-1] in double precision.
- real(sp), parameter mo_constants::solarconst_sp = 1367._sp

  Solar constant in [J m^-2 s^-1] in single precision.
- real(dp), parameter mo_constants::specheatet_dp = 2.45e06_dp

  Specific heat for vaporization of water in [J m-2 mm-1] in double precision.
- real(sp), parameter mo_constants::specheatet_sp = 2.45e06_sp

  Specific heat for vaporization of water in [J m-2 mm-1] in single precision.
- real(dp), parameter mo_constants::t0_dp = 273.15_dp

  Standard temperature [K] in double precision.
- real(sp), parameter mo_constants::t0_sp = 273.15_sp

  Standard temperature [K] in single precision.
- real(dp), parameter mo_constants::sigma_dp = 5.67e-08_dp

  Stefan-Boltzmann constant [W m^-2 K^-4] in double precision.
- real(sp), parameter mo_constants::sigma_sp = 5.67e-08_sp

  Stefan-Boltzmann constant [W m^-2 K^-4] in single precision.
- real(sp), parameter mo_constants::radiusearth_sp = 6371228._sp
- real(dp), parameter mo_constants::radiusearth_dp = 6371228._dp
- real(dp), parameter mo_constants::p0_dp = 101325._dp

  standard atmospehere Standard pressure [Pa] in double precision
- real(sp), parameter mo_constants::p0_sp = 101325._sp

  Standard pressure [Pa] in single precision.
- real(dp), parameter mo_constants::rho0_dp = 1.225_dp

  standard density [kg m^-3] in double precision
- real(sp), parameter mo_constants::rho0_sp = 1.225_sp

  standard density [kg m^-3] in single precision
- real(dp), parameter mo_constants::cp0_dp = 1005.0_dp

  specific heat capacity of air [J kg^-1 K^-1] in double precision
- real(sp), parameter mo_constants::cp0_sp = 1005.0_sp

  specific heat capacity of air [J kg^-1 K^-1] in single precision
- real(dp), parameter mo_constants::pi_d = 3.14159265358979323846264338327950288419 7_dp

  Pi in double precision.
- real(sp), parameter mo_constants::pi = 3.14159265358979323846264338327950288419 7_sp

  Pi in single precision.

- real(dp), parameter mo_constants::pio2_d = 1.57079632679489661923132169163975144209858_dp

  Pi/2 in double precision.
- real(sp), parameter mo_constants::pio2 = 1.57079632679489661923132169163975144209858_sp

  Pi/2 in single precision.
- real(dp), parameter mo_constants::twopi_d = 6.28318530717958647692528676655900576839 4_dp

  2∗Pi in double precision
- real(sp), parameter mo_constants::twopi = 6.28318530717958647692528676655900576839 4_sp

  2∗Pi in single precision
- real(dp), parameter mo_constants::sqrt2_d = 1.41421356237309504880168872420969807856967_dp

  Square root of 2 in double precision.
- real(sp), parameter mo_constants::sqrt2 = 1.41421356237309504880168872420969807856967_sp

  Square root of 2 in single precision.
- real(dp), parameter mo_constants::euler_d = 0.57721566490153286060651209008240243104 22_dp

  Euler"s constant in double precision.
- real(sp), parameter mo_constants::euler = 0.57721566490153286060651209008240243104 22_sp

  Euler"s constant in single precision.
- integer, parameter mo_constants::nin = input_unit

  Standard input file unit.
- integer, parameter mo_constants::nout = output_unit

  Standard output file unit.
- integer, parameter mo_constants::nerr = error_unit

  Standard error file unit.
- integer, parameter mo_constants::nnml = 100

  Standard file unit for namelist.

18.27 mo_corr.f90 File Reference

Data Types

- interface mo_corr::autocoeffk
- interface mo_corr::autocorr
- interface mo_corr::corr
- interface mo_corr::crosscoeffk
- interface mo_corr::crosscorr
- interface mo_corr::arth
- interface mo_corr::four1
- interface mo_corr::fourrow
- interface mo_corr::realft
- interface mo_corr::swap

Modules

- module mo_corr

Functions/Subroutines

- real(sp) function, dimension(n) mo_corr::arth_sp (first, increment, n)
- real(dp) function, dimension(n) mo_corr::arth_dp (first, increment, n)
- integer(i4) function, dimension(n) mo_corr::arth_i4 (first, increment, n)
- real(dp) function mo_corr::autocoeffk_dp (x, k, mask)
- real(sp) function mo_corr::autocoeffk_sp (x, k, mask)
- real(dp) function, dimension(size(k)) mo_corr::autocoeffk_1d_dp (x, k, mask)
- real(sp) function, dimension(size(k)) mo_corr::autocoeffk_1d_sp (x, k, mask)
- real(dp) function mo_corr::autocorr_dp (x, k, mask)
- real(sp) function mo_corr::autocorr_sp (x, k, mask)
- real(dp) function, dimension(size(k)) mo_corr::autocorr_1d_dp (x, k, mask)
- real(sp) function, dimension(size(k)) mo_corr::autocorr_1d_sp (x, k, mask)
- real(dp) function, dimension(size(data1)) mo_corr::corr_dp (data1, data2, nadjust, nhigh, nwin)
- real(sp) function, dimension(size(data1)) mo_corr::corr_sp (data1, data2, nadjust, nhigh, nwin)
- real(dp) function mo_corr::crosscoeffk_dp (x, y, k, mask)
- real(sp) function mo_corr::crosscoeffk_sp (x, y, k, mask)
- real(dp) function mo_corr::crosscorr_dp (x, y, k, mask)
- real(sp) function mo_corr::crosscorr_sp (x, y, k, mask)
- subroutine mo_corr::four1_sp (data, isign)
- subroutine mo_corr::four1_dp (data, isign)
- subroutine mo_corr::fourrow_sp (data, isign)
- subroutine mo_corr::fourrow_dp (data, isign)
- subroutine mo_corr::realft_dp (data, isign, zdata)
- subroutine mo_corr::realft_sp (data, isign, zdata)
- subroutine mo_corr::swap_1d_spc (a, b)
- subroutine mo_corr::swap_1d_dpc (a, b)
- complex(dpc) function, dimension(nn) mo_corr::zroots_unity_dp (n, nn)
- complex(spc) function, dimension(nn) mo_corr::zroots_unity_sp (n, nn)

Variables

- integer(i4), parameter mo_corr::npar_arth = 16
- integer(i4), parameter mo_corr::npar2_arth = 8

18.28 mo_dds.f90 File Reference

Modules

- module mo_dds

  Dynamically Dimensioned Search (DDS)

Functions/Subroutines

- real(dp) function, dimension(size(pini)), public mo_dds::dds (eval, obj_func, pini, prange, r, seed, maxiter, maxit, mask, tmp_file, funcbest, history)

  DDS.
- real(dp) function, dimension(size(pini)), public mo_dds::mdds (eval, obj_func, pini, prange, seed, maxiter, maxit, mask, tmp_file, funcbest, history)

  MDDS.
- subroutine mo_dds::neigh_value (x_cur, x_min, x_max, r, new_value)

18.29 mo_errormeasures.f90 File Reference

Data Types

- interface mo_errormeasures::bias
- interface mo_errormeasures::kge

  Kling-Gupta-Efficiency measure.
- interface mo_errormeasures::kgenocorr

  Kling-Gupta-Efficiency measure without correlation.
- interface mo_errormeasures::lnnse
- interface mo_errormeasures::mae
- interface mo_errormeasures::mse
- interface mo_errormeasures::nse
- interface mo_errormeasures::sae
- interface mo_errormeasures::sse
- interface mo_errormeasures::rmse
- interface mo_errormeasures::wnse

Modules

- module mo_errormeasures

Functions/Subroutines

- real(sp) function mo_errormeasures::bias_sp_1d (x, y, mask)
- real(dp) function mo_errormeasures::bias_dp_1d (x, y, mask)
- real(sp) function mo_errormeasures::bias_sp_2d (x, y, mask)
- real(dp) function mo_errormeasures::bias_dp_2d (x, y, mask)
- real(sp) function mo_errormeasures::bias_sp_3d (x, y, mask)
- real(dp) function mo_errormeasures::bias_dp_3d (x, y, mask)
- real(sp) function mo_errormeasures::kge_sp_1d (x, y, mask)
- real(sp) function mo_errormeasures::kge_sp_2d (x, y, mask)
- real(sp) function mo_errormeasures::kge_sp_3d (x, y, mask)
- real(dp) function mo_errormeasures::kge_dp_1d (x, y, mask)
- real(dp) function mo_errormeasures::kge_dp_2d (x, y, mask)
- real(dp) function mo_errormeasures::kge_dp_3d (x, y, mask)
- real(sp) function mo_errormeasures::kgenocorr_sp_1d (x, y, mask)
- real(sp) function mo_errormeasures::kgenocorr_sp_2d (x, y, mask)
- real(sp) function mo_errormeasures::kgenocorr_sp_3d (x, y, mask)
- real(dp) function mo_errormeasures::kgenocorr_dp_1d (x, y, mask)
- real(dp) function mo_errormeasures::kgenocorr_dp_2d (x, y, mask)
- real(dp) function mo_errormeasures::kgenocorr_dp_3d (x, y, mask)
- real(sp) function mo_errormeasures::lnnse_sp_1d (x, y, mask)
- real(dp) function mo_errormeasures::lnnse_dp_1d (x, y, mask)
- real(sp) function mo_errormeasures::lnnse_sp_2d (x, y, mask)
- real(dp) function mo_errormeasures::lnnse_dp_2d (x, y, mask)
- real(sp) function mo_errormeasures::lnnse_sp_3d (x, y, mask)
- real(dp) function mo_errormeasures::lnnse_dp_3d (x, y, mask)
- real(sp) function mo_errormeasures::mae_sp_1d (x, y, mask)
- real(dp) function mo_errormeasures::mae_dp_1d (x, y, mask)
- real(sp) function mo_errormeasures::mae_sp_2d (x, y, mask)
- real(dp) function mo_errormeasures::mae_dp_2d (x, y, mask)
- real(sp) function mo_errormeasures::mae_sp_3d (x, y, mask)

- real(dp) function mo_errormeasures::mae_dp_3d (x, y, mask)
- real(sp) function mo_errormeasures::mse_sp_1d (x, y, mask)
- real(dp) function mo_errormeasures::mse_dp_1d (x, y, mask)
- real(sp) function mo_errormeasures::mse_sp_2d (x, y, mask)
- real(dp) function mo_errormeasures::mse_dp_2d (x, y, mask)
- real(sp) function mo_errormeasures::mse_sp_3d (x, y, mask)
- real(dp) function mo_errormeasures::mse_dp_3d (x, y, mask)
- real(sp) function mo_errormeasures::nse_sp_1d (x, y, mask)
- real(dp) function mo_errormeasures::nse_dp_1d (x, y, mask)
- real(sp) function mo_errormeasures::nse_sp_2d (x, y, mask)
- real(dp) function mo_errormeasures::nse_dp_2d (x, y, mask)
- real(sp) function mo_errormeasures::nse_sp_3d (x, y, mask)
- real(dp) function mo_errormeasures::nse_dp_3d (x, y, mask)
- real(sp) function mo_errormeasures::sae_sp_1d (x, y, mask)
- real(dp) function mo_errormeasures::sae_dp_1d (x, y, mask)
- real(sp) function mo_errormeasures::sae_sp_2d (x, y, mask)
- real(dp) function mo_errormeasures::sae_dp_2d (x, y, mask)
- real(sp) function mo_errormeasures::sae_sp_3d (x, y, mask)
- real(dp) function mo_errormeasures::sae_dp_3d (x, y, mask)
- real(sp) function mo_errormeasures::sse_sp_1d (x, y, mask)
- real(dp) function mo_errormeasures::sse_dp_1d (x, y, mask)
- real(sp) function mo_errormeasures::sse_sp_2d (x, y, mask)
- real(dp) function mo_errormeasures::sse_dp_2d (x, y, mask)
- real(sp) function mo_errormeasures::sse_sp_3d (x, y, mask)
- real(dp) function mo_errormeasures::sse_dp_3d (x, y, mask)
- real(sp) function mo_errormeasures::rmse_sp_1d (x, y, mask)
- real(dp) function mo_errormeasures::rmse_dp_1d (x, y, mask)
- real(sp) function mo_errormeasures::rmse_sp_2d (x, y, mask)
- real(dp) function mo_errormeasures::rmse_dp_2d (x, y, mask)
- real(sp) function mo_errormeasures::rmse_sp_3d (x, y, mask)
- real(dp) function mo_errormeasures::rmse_dp_3d (x, y, mask)
- real(sp) function mo_errormeasures::wnse_sp_1d (x, y, mask)
- real(dp) function mo_errormeasures::wnse_dp_1d (x, y, mask)
- real(sp) function mo_errormeasures::wnse_sp_2d (x, y, mask)
- real(dp) function mo_errormeasures::wnse_dp_2d (x, y, mask)
- real(sp) function mo_errormeasures::wnse_sp_3d (x, y, mask)
- real(dp) function mo_errormeasures::wnse_dp_3d (x, y, mask)

18.30 mo_file.f90 File Reference

Modules

- module mo_file

  Provides file names and units for mHM.

Variables

- character(len=∗), parameter mo_file::version = '5.10'

  Current mHM model version.
- character(len=∗), parameter mo_file::version_date = 'June 2019'

  Time of current mHM model version release.
- character(len=∗), parameter mo_file::file_main = 'mhm_driver.f90'

Driver file.

- character(len=*), parameter mo_file::file_namelist_mhm = 'mhm.nml'

  Namelist file name.
- integer, parameter mo_file::unamelist_mhm = 30

  Unit for namelist.
- character(len=*), parameter mo_file::file_namelist_mhm_param = 'mhm_parameter.nml'

  Parameter namelists file name.
- integer, parameter mo_file::unamelist_mhm_param = 31

  Unit for namelist.
- character(len=*), parameter mo_file::file_defoutput = 'mhm_outputs.nml'

  file defining mHM's outputs
- integer, parameter mo_file::udefoutput = 67

  Unit for file defining mHM's outputs.
- integer, parameter mo_file::utws = 77

  unit for tws time series

18.31 mo_finish.f90 File Reference

Modules

- module mo_finish

  Finish a program gracefully.

Functions/Subroutines

- subroutine, public mo_finish::finish (name, text, unit)

  Finish a program gracefully.

18.32 mo_global_variables.f90 File Reference

Data Types

- type mo_global_variables::twsstructure

Modules

- module mo_global_variables

  Global variables ONLY used in reading, writing and startup.

Variables

- integer(i4) mo_global_variables::timestep_model_outputs
- logical, dimension(noutflxstate) mo_global_variables::outputflxstate
- integer(i4), dimension(:), allocatable, public mo_global_variables::timestep_model_inputs
- logical, public mo_global_variables::read_meteo_weights
- character(256), public mo_global_variables::inputformat_meteo_forcings
- integer(i4), public mo_global_variables::timestep_sm_input
- integer(i4), public mo_global_variables::timestep_neutrons_input
- integer(i4), public mo_global_variables::timestep_et_input

- character(256), dimension(:), allocatable, public mo_global_variables::dirprecipitation
- character(256), dimension(:), allocatable, public mo_global_variables::dirtemperature
- character(256), dimension(:), allocatable, public mo_global_variables::dirmintemperature
- character(256), dimension(:), allocatable, public mo_global_variables::dirmaxtemperature
- character(256), dimension(:), allocatable, public mo_global_variables::dirnetradiation
- character(256), dimension(:), allocatable, public mo_global_variables::dirabsvappressure
- character(256), dimension(:), allocatable, public mo_global_variables::dirwindspeed
- character(256), dimension(:), allocatable, public mo_global_variables::dirreferenceet
- character(256), dimension(:), allocatable, public mo_global_variables::dirsoil_moisture
- character(256), dimension(:), allocatable, public mo_global_variables::filetws
- character(256), dimension(:), allocatable, public mo_global_variables::dirneutrons
- character(256), dimension(:), allocatable, public mo_global_variables::direvapotranspiration
- integer(i4), parameter, public mo_global_variables::routingstates = 2
- type(twsstructure), public mo_global_variables::basin_avg_tws_obs
- real(dp), dimension(:, :), allocatable, public mo_global_variables::basin_avg_tws_sim
- integer(i4), public mo_global_variables::nmeasperday_tws
- type(grid), dimension(:), allocatable, public mo_global_variables::level2
- real(dp), dimension(:, :, :), allocatable, public mo_global_variables::l1_temp_weights
- real(dp), dimension(:, :, :), allocatable, public mo_global_variables::l1_pet_weights
- real(dp), dimension(:, :, :), allocatable, public mo_global_variables::l1_pre_weights
- real(dp), dimension(:, :), allocatable, public mo_global_variables::l1_pre
- real(dp), dimension(:, :), allocatable, public mo_global_variables::l1_temp
- real(dp), dimension(:, :), allocatable, public mo_global_variables::l1_pet
- real(dp), dimension(:, :), allocatable, public mo_global_variables::l1_tmin
- real(dp), dimension(:, :), allocatable, public mo_global_variables::l1_tmax
- real(dp), dimension(:, :), allocatable, public mo_global_variables::l1_netrad
- real(dp), dimension(:, :), allocatable, public mo_global_variables::l1_absvappress
- real(dp), dimension(:, :), allocatable, public mo_global_variables::l1_windspeed
- real(dp), dimension(:, :), allocatable, public mo_global_variables::l1_sm
- logical, dimension(:, :), allocatable, public mo_global_variables::l1_sm_mask
- integer(i4) mo_global_variables::ntimesteps_l1_sm
- integer(i4) mo_global_variables::nsoilhorizons_sm_input
- real(dp), dimension(:, :), allocatable, public mo_global_variables::l1_neutronsdata
- logical, dimension(:, :), allocatable, public mo_global_variables::l1_neutronsdata_mask
- integer(i4) mo_global_variables::ntimesteps_l1_neutrons
- real(dp), dimension(:, :), allocatable, public mo_global_variables::l1_et
- logical, dimension(:, :), allocatable, public mo_global_variables::l1_et_mask
- integer(i4) mo_global_variables::ntimesteps_l1_et
- real(dp), dimension(:), allocatable, public mo_global_variables::l1_inter
- real(dp), dimension(:), allocatable, public mo_global_variables::l1_snowpack
- real(dp), dimension(:), allocatable, public mo_global_variables::l1_sealstw
- real(dp), dimension(:, :), allocatable, public mo_global_variables::l1_soilmoist
- real(dp), dimension(:), allocatable, public mo_global_variables::l1_unsatstw
- real(dp), dimension(:), allocatable, public mo_global_variables::l1_satstw
- real(dp), dimension(:), allocatable, public mo_global_variables::l1_neutrons
- real(dp), dimension(:), allocatable, public mo_global_variables::l1_pet_calc
- real(dp), dimension(:, :), allocatable, public mo_global_variables::l1_aetsoil
- real(dp), dimension(:), allocatable, public mo_global_variables::l1_aetcanopy
- real(dp), dimension(:), allocatable, public mo_global_variables::l1_aetsealed
- real(dp), dimension(:), allocatable, public mo_global_variables::l1_baseflow
- real(dp), dimension(:, :), allocatable, public mo_global_variables::l1_infilsoil
- real(dp), dimension(:), allocatable, public mo_global_variables::l1_fastrunoff
- real(dp), dimension(:), allocatable, public mo_global_variables::l1_melt
- real(dp), dimension(:), allocatable, public mo_global_variables::l1_percol
- real(dp), dimension(:), allocatable, public mo_global_variables::l1_preeffect

- real(dp), dimension(:), allocatable, public mo_global_variables::l1_rain
- real(dp), dimension(:), allocatable, public mo_global_variables::l1_runoffseal
- real(dp), dimension(:), allocatable, public mo_global_variables::l1_slowrunoff
- real(dp), dimension(:), allocatable, public mo_global_variables::l1_snow
- real(dp), dimension(:), allocatable, public mo_global_variables::l1_throughfall
- real(dp), dimension(:), allocatable, public mo_global_variables::l1_total_runoff
- real(dp), dimension(int(yearmonths, i4)), public mo_global_variables::evap_coeff
- real(dp), dimension(int(yearmonths, i4)), public mo_global_variables::fday_prec
- real(dp), dimension(int(yearmonths, i4)), public mo_global_variables::fnight_prec
- real(dp), dimension(int(yearmonths, i4)), public mo_global_variables::fday_pet
- real(dp), dimension(int(yearmonths, i4)), public mo_global_variables::fnight_pet
- real(dp), dimension(int(yearmonths, i4)), public mo_global_variables::fday_temp
- real(dp), dimension(int(yearmonths, i4)), public mo_global_variables::fnight_temp
- real(dp), dimension(:), allocatable, public mo_global_variables::neutron_integral_afast

18.33 mo_grid.f90 File Reference

Modules

- module mo_grid

  TODO: add description.

Functions/Subroutines

- subroutine, public mo_grid::init_lowres_level (highres, target_resolution, lowres, highres_lowres_remap)

  Level-1 variable initialization.
- subroutine, public mo_grid::set_basin_indices (grids)

  TODO: add description.
- subroutine, public mo_grid::l0_grid_setup (new_grid)

  level 0 variable initialization
- subroutine, public mo_grid::mapcoordinates (level, y, x)

  Generate map coordinates.
- subroutine, public mo_grid::geocoordinates (level, lat, lon)

  Generate geographic coordinates.
- subroutine mo_grid::calculate_grid_properties (nrowsIn, ncolsIn, xllcornerIn, yllcornerIn, cellsizeIn, aiming↩ Resolution, nrowsOut, ncolsOut, xllcornerOut, yllcornerOut, cellsizeOut)

  Calculates basic grid properties at a required coarser level using information of a given finer level. Calculates basic grid properties at a required coarser level (e.g., L11) using information of a given finer level (e.g., L0). Basic grid properties such as nrows, ncols, xllcorner, yllcorner cellsize are estimated in this routine.

18.34 mo_init_states.f90 File Reference

Modules

- module mo_init_states

  Initialization of all state variables of mHM.

Functions/Subroutines

- subroutine, public mo_init_states::variables_alloc (ncells1)

  Allocation of space for mHM related L1 and L11 variables.
- subroutine, public mo_init_states::variables_default_init

  Default initalization mHM related L1 variables.

18.35 mo_julian.f90 File Reference

Data Types

- interface mo_julian::setcalendar

Modules

- module mo_julian

  Julian date conversion routines.

Functions/Subroutines

- subroutine mo_julian::setcalendarstring (selector)

  Set module private variable calendar.
- subroutine mo_julian::setcalendarinteger (selector)

  Set module private variable calendar.
- pure integer(i4) function mo_julian::selectcalendar (selector)

  Select a calendar.
- elemental subroutine, public mo_julian::caldat (julian, dd, mm, yy, calendar)

  Day, month and year from Julian day in the current or given calendar.
- elemental subroutine, public mo_julian::dec2date (julian, dd, mm, yy, hh, nn, ss, calendar)

  Day, month, year, hour, minute, and second from fractional Julian day in the current or given calendar.
- elemental real(dp) function, public mo_julian::date2dec (dd, mm, yy, hh, nn, ss, calendar)

  Fractional Julian day from day, month, year, hour, minute, second in the current calendar.
- elemental integer(i4) function, public mo_julian::julday (dd, mm, yy, calendar)

  Julian day from day, month and year in the current or given calendar.
- elemental subroutine, public mo_julian::caldatjulian (julian, dd, mm, yy)

  Day, month and year from Julian day.
- elemental real(dp) function mo_julian::date2decjulian (dd, mm, yy, hh, nn, ss)

  Fractional Julian day from day, month, year, hour, minute, second.
- elemental subroutine mo_julian::dec2datejulian (julian, dd, mm, yy, hh, nn, ss)

  Day, month, year, hour, minute, and second from fractional Julian day.
- elemental integer(i4) function mo_julian::juldayjulian (dd, mm, yy)

  Julian day from day, month and year.
- elemental integer(i4) function, public mo_julian::ndays (dd, mm, yy)

  IMSL Julian day from day, month and year.
- elemental subroutine, public mo_julian::ndyin (julian, dd, mm, yy)

  Day, month and year from IMSL Julian day.
- elemental subroutine mo_julian::caldat360 (julian, dd, mm, yy)

  Day, month and year from Julian day in a 360 day calendar.
- elemental integer(i4) function mo_julian::julday360 (dd, mm, yy)

  Julian day from day, month and year in a 360_day calendar.

- elemental subroutine mo_julian::dec2date360 (julian, dd, mm, yy, hh, nn, ss)

  Day, month, year, hour, minute, and second from fractional Julian day in a 360_day calendar.
- elemental real(dp) function mo_julian::date2dec360 (dd, mm, yy, hh, nn, ss)

  Fractional Julian day from day, month, year, hour, minute, second in 360 day calendar.
- elemental subroutine mo_julian::caldat365 (julian, dd, mm, yy)

  Day, month and year from Julian day in a 365 day calendar.
- elemental integer(i4) function mo_julian::julday365 (dd, mm, yy)

  Julian day from day, month and year in a 365_day calendar.
- elemental subroutine mo_julian::dec2date365 (julian, dd, mm, yy, hh, nn, ss)

  Day, month, year, hour, minute, and second from fractional Julian day in a 365_day calendar.
- elemental real(dp) function mo_julian::date2dec365 (dd, mm, yy, hh, nn, ss)

  Fractional Julian day from day, month, year, hour, minute, second in 365 day calendar.

Variables

- integer(i4), save, private mo_julian::calendar = 1

18.36 mo_kind.f90 File Reference

Data Types

- type mo_kind::sprs2_sp

  Single Precision Numerical Recipes types for sparse arrays.
- type mo_kind::sprs2_dp

  Double Precision Numerical Recipes types for sparse arrays.

Modules

- module mo_kind

  Define number representations.

Variables

- integer, parameter mo_kind::i1 = SELECTED_INT_KIND(2)

  1 Byte Integer Kind
- integer, parameter mo_kind::i2 = c_short

  2 Byte Integer Kind
- integer, parameter mo_kind::i4 = c_int

  4 Byte Integer Kind
- integer, parameter mo_kind::i8 = c_long_long

  8 Byte Integer Kind
- integer, parameter mo_kind::sp = c_float

  Single Precision Real Kind.
- integer, parameter mo_kind::dp = c_double

  Double Precision Real Kind.
- integer, parameter mo_kind::spc = c_float_complex

  Single Precision Complex Kind.
- integer, parameter mo_kind::dpc = c_double_complex

  Double Precision Complex Kind.
- integer, parameter mo_kind::lgt = KIND(.true.)

  Logical Kind.

18.37 mo_linfit.f90 File Reference

Data Types

- interface mo_linfit::linfit

  Fits a straight line to input data by minimizing chi^2.

Modules

- module mo_linfit

  Fitting a straight line.

Functions/Subroutines

- real(dp) function, dimension(:), allocatable mo_linfit::linfit_dp (x, y, a, b, siga, sigb, chi2, model2)
- real(sp) function, dimension(:), allocatable mo_linfit::linfit_sp (x, y, a, b, siga, sigb, chi2, model2)

18.38 mo_mad.f90 File Reference

Data Types

- interface mo_mad::mad

Modules

- module mo_mad

Functions/Subroutines

- logical function, dimension(size(arr)) mo_mad::mad_dp (arr, z, mask, deriv)
- logical function, dimension(size(arr)) mo_mad::mad_sp (arr, z, mask, deriv)
- real(dp) function, dimension(size(arr)) mo_mad::mad_val_dp (arr, z, mask, tout, mval)
- real(sp) function, dimension(size(arr)) mo_mad::mad_val_sp (arr, z, mask, tout, mval)

18.39 mo_mcmc.f90 File Reference

Data Types

- interface mo_mcmc::mcmc

  This module is Monte Carlo Markov Chain sampling of a posterior parameter distribution (measurement errors are either known or modeled).
- interface mo_mcmc::mcmc_stddev

  This module is Monte Carlo Markov Chain sampling of a posterior parameter distribution (measurement errors are neither known nor modeled).

Modules

- module mo_mcmc

  This module is Monte Carlo Markov Chain sampling of a posterior parameter distribution.

Functions/Subroutines

- subroutine mo_mcmc::mcmc_dp (eval, likelihood, para, rangePar, mcmc_paras, burnin_paras, seed_in, printflag_in, maskpara_in, restart, restart_file, tmp_file, loglike_in, ParaSelectMode_in, iter_burnin_in, iter_↩ mcmc_in, chains_in, stepsize_in)
- subroutine mo_mcmc::mcmc_stddev_dp (eval, likelihood, para, rangePar, mcmc_paras, burnin_paras, seed_in, printflag_in, maskpara_in, tmp_file, loglike_in, ParaSelectMode_in, iter_burnin_in, iter_mcmc_in, chains_in, stepsize_in)
- real(dp) function mo_mcmc::pargen_dp (old, dMax, oMin, oMax, RN, inbound)
- real(dp) function mo_mcmc::pargennorm_dp (old, dMax, oMin, oMax, RN, inbound)
- recursive subroutine mo_mcmc::generatenewparameterset_dp (ParaSelectMode, paraold, truepara, range↩ Par, stepsize, save_state_2, save_state_3, paranew, ChangePara)

18.40 mo_message.f90 File Reference

Modules

- module mo_message

  Write out concatenated strings.

Functions/Subroutines

- subroutine, public mo_message::message (t01, t02, t03, t04, t05, t06, t07, t08, t09, t10, uni, advance)

  Write out several string concatenated either on screen or in a file.

Variables

- character(len=1024), public mo_message::message_text = ''

18.41 mo_meteo_forcings.f90 File Reference

Modules

- module mo_meteo_forcings

  Prepare meteorological forcings data for mHM.

Functions/Subroutines

- subroutine, public mo_meteo_forcings::prepare_meteo_forcings_data (iBasin, tt)

  Prepare meteorological forcings data for a given variable.
- subroutine mo_meteo_forcings::meteo_forcings_wrapper (iBasin, dataPath, inputFormat, dataOut1, lower, upper, ncvarName)

  Prepare meteorological forcings data for mHM at Level-1.
- subroutine mo_meteo_forcings::meteo_weights_wrapper (iBasin, read_meteo_weights, dataPath, dataOut1, lower, upper, ncvarName)

  Prepare weights for meteorological forcings data for mHM at Level-1.
- subroutine mo_meteo_forcings::chunk_config (iBasin, tt, read_flag, readPer)

  determines the start date, end date, and read_flag given basin id and current timestep
- logical function mo_meteo_forcings::is_read (iBasin, tt)

  evaluate whether new chunk should be read at this timestep

- subroutine mo_meteo_forcings::chunk_size (iBasin, tt, readPer)

  calculate beginning and end of read Period, i.e. that is length of current chunk to read

18.42 mo_mhm.f90 File Reference

Modules

- module mo_mhm

  Call all main processes of mHM.

Functions/Subroutines

- subroutine, public mo_mhm::mhm (read_states, tt, time, processMatrix, horizon_depth, nCells1, nHorizons↩ _mHM, ntimesteps_day, c2TSTu, neutron_integral_AFast, global_parameters, latitude, evap_coeff, fday_↩ prec, fnight_prec, fday_pet, fnight_pet, fday_temp, fnight_temp, temp_weights, pet_weights, pre_weights, read_meteo_weights, pet_in, tmin_in, tmax_in, netrad_in, absvappres_in, windspeed_in, prec_in, temp_in, fSealed1, interc, snowpack, sealedStorage, soilMoisture, unsatStorage, satStorage, neutrons, pet_calc, aet↩ _soil, aet_canopy, aet_sealed, baseflow, infiltration, fast_interflow, melt, perc, prec_effect, rain, runoff_sealed, slow_interflow, snow, throughfall, total_runoff, alpha, deg_day_incr, deg_day_max, deg_day_noprec, deg↩ _day, fAsp, petLAIcorFactorL1, HarSamCoeff, PrieTayAlpha, aeroResist, surfResist, frac_roots, interc_max, karst_loss, k0, k1, k2, kp, soil_moist_FC, soil_moist_sat, soil_moist_exponen, jarvis_thresh_c1, temp_thresh, unsat_thresh, water_thresh_sealed, wilting_point)

  Pure mHM calculations.

18.43 mo_mhm_constants.f90 File Reference

Modules

- module mo_mhm_constants

  Provides mHM specific constants.

Variables

- real(dp), parameter, public mo_mhm_constants::h2odens = 1000.0_dp
- real(dp), parameter, public mo_mhm_constants::p2_initstatefluxes = 15.00_dp
- real(dp), parameter, public mo_mhm_constants::p3_initstatefluxes = 10.00_dp
- real(dp), parameter, public mo_mhm_constants::p4_initstatefluxes = 75.00_dp
- real(dp), parameter, public mo_mhm_constants::p5_initstatefluxes = 1500.00_dp
- real(dp), parameter, public mo_mhm_constants::c1_initstatesm = 0.25_dp
- integer(i4), parameter, public mo_mhm_constants::noutflxstate = 20_i4
- real(dp), parameter, public mo_mhm_constants::stboltzmann = 5.67e-08_dp

  Stefan-Boltzmann constant [W m^-2 K^-4].
- real(dp), parameter, public mo_mhm_constants::harsamconst = 17.800_dp

  Hargreaves-Samani ref. ET formula [deg C].
- real(dp), parameter, public mo_mhm_constants::duffiedr = 0.0330_dp
- real(dp), parameter, public mo_mhm_constants::duffiedelta1 = 0.4090_dp
- real(dp), parameter, public mo_mhm_constants::duffiedelta2 = 1.3900_dp
- real(dp), parameter, public mo_mhm_constants::tetens_c1 = 0.6108_dp

  Tetens's formula to calculate saturated vapour pressure.
- real(dp), parameter, public mo_mhm_constants::tetens_c2 = 17.270_dp

- real(dp), parameter, public mo_mhm_constants::tetens_c3 = 237.30_dp
- real(dp), parameter, public mo_mhm_constants::satpressureslope1 = 4098.0_dp

  calculation of the slope of the saturation vapour pressure curve following Tetens
- real(dp), parameter, public mo_mhm_constants::desilets_a0 = 0.0808_dp

  Neutrons and moisture: N0 formula, Desilets et al. 2010.
- real(dp), parameter, public mo_mhm_constants::desilets_a1 = 0.372_dp
- real(dp), parameter, public mo_mhm_constants::desilets_a2 = 0.115_dp
- real(dp), parameter, public mo_mhm_constants::cosmic_bd = 1.4020_dp

  Neutrons and moisture: COSMIC, Shuttleworth et al. 2013.
- real(dp), parameter, public mo_mhm_constants::cosmic_vwclat = 0.0753_dp
- real(dp), parameter, public mo_mhm_constants::cosmic_n = 348.33_dp
- real(dp), parameter, public mo_mhm_constants::cosmic_alpha = 0.2392421548_dp
- real(dp), parameter, public mo_mhm_constants::cosmic_l1 = 161.98621864_dp
- real(dp), parameter, public mo_mhm_constants::cosmic_l2 = 129.14558985_dp
- real(dp), parameter, public mo_mhm_constants::cosmic_l3 = 107.82204562_dp
- real(dp), parameter, public mo_mhm_constants::cosmic_l4 = 3.1627190566_dp

18.44 mo_mhm_eval.f90 File Reference

Modules

- module mo_mhm_eval

  Runs mhm with a specific parameter set and returns required variables, e.g. runoff.

Functions/Subroutines

- subroutine, public mo_mhm_eval::mhm_eval (parameterset, runoff, sm_opti, basin_avg_tws, neutrons_opti, et_opti)

  Runs mhm with a specific parameter set and returns required variables, e.g. runoff.

18.45 mo_mhm_read_config.f90 File Reference

Modules

- module mo_mhm_read_config

  Reading of main model configurations.

Functions/Subroutines

- subroutine, public mo_mhm_read_config::mhm_read_config (file_namelist, unamelist)

  Read main configurations for mHM.

18.46 mo_moment.f90 File Reference

Data Types

- interface mo_moment::absdev
- interface mo_moment::average
- interface mo_moment::central_moment

- interface mo_moment::central_moment_var
- interface mo_moment::correlation
- interface mo_moment::covariance
- interface mo_moment::kurtosis
- interface mo_moment::mean
- interface mo_moment::mixed_central_moment
- interface mo_moment::mixed_central_moment_var
- interface mo_moment::moment
- interface mo_moment::skewness
- interface mo_moment::stddev
- interface mo_moment::variance

Modules

- module mo_moment

Functions/Subroutines

- real(dp) function mo_moment::absdev_dp (dat, mask)
- real(sp) function mo_moment::absdev_sp (dat, mask)
- real(dp) function mo_moment::average_dp (dat, mask)
- real(sp) function mo_moment::average_sp (dat, mask)
- real(dp) function mo_moment::central_moment_dp (x, r, mask)
- real(sp) function mo_moment::central_moment_sp (x, r, mask)
- real(dp) function mo_moment::central_moment_var_dp (x, r, mask)
- real(sp) function mo_moment::central_moment_var_sp (x, r, mask)
- real(dp) function mo_moment::correlation_dp (x, y, mask)
- real(sp) function mo_moment::correlation_sp (x, y, mask)
- real(dp) function mo_moment::covariance_dp (x, y, mask)
- real(sp) function mo_moment::covariance_sp (x, y, mask)
- real(dp) function mo_moment::kurtosis_dp (dat, mask)
- real(sp) function mo_moment::kurtosis_sp (dat, mask)
- real(dp) function mo_moment::mean_dp (dat, mask)
- real(sp) function mo_moment::mean_sp (dat, mask)
- real(dp) function mo_moment::mixed_central_moment_dp (x, y, r, s, mask)
- real(sp) function mo_moment::mixed_central_moment_sp (x, y, r, s, mask)
- real(dp) function mo_moment::mixed_central_moment_var_dp (x, y, r, s, mask)
- real(sp) function mo_moment::mixed_central_moment_var_sp (x, y, r, s, mask)
- subroutine mo_moment::moment_dp (dat, average, variance, skewness, kurtosis, mean, stddev, absdev, mask)
- subroutine mo_moment::moment_sp (dat, average, variance, skewness, kurtosis, mean, stddev, absdev, mask)
- real(dp) function mo_moment::stddev_dp (dat, mask)
- real(sp) function mo_moment::stddev_sp (dat, mask)
- real(dp) function mo_moment::skewness_dp (dat, mask)
- real(sp) function mo_moment::skewness_sp (dat, mask)
- real(dp) function mo_moment::variance_dp (dat, mask)
- real(sp) function mo_moment::variance_sp (dat, mask)

18.47 mo_mpr_constants.f90 File Reference

Modules

- module mo_mpr_constants

  Provides MPR specific constants.

Variables

- integer(i4), parameter, public mo_mpr_constants::nlcover_class = 3_i4
- integer(i4), parameter, public mo_mpr_constants::maxgeounit = 25_i4
- integer(i4), parameter, public mo_mpr_constants::maxnosoilhorizons = 10_i4
- real(dp), parameter, public mo_mpr_constants::p2_initstatefluxes = 15.00_dp
- real(dp), parameter, public mo_mpr_constants::p3_initstatefluxes = 10.00_dp
- real(dp), parameter, public mo_mpr_constants::p4_initstatefluxes = 75.00_dp
- real(dp), parameter, public mo_mpr_constants::p5_initstatefluxes = 1500.00_dp
- real(dp), parameter, public mo_mpr_constants::c1_initstatesm = 0.25_dp
- real(dp), parameter, public mo_mpr_constants::bulkdens_orgmatter = 0.224_dp
- real(dp), parameter, public mo_mpr_constants::field_cap_c1 = -0.60_dp
- real(dp), parameter, public mo_mpr_constants::field_cap_c2 = 2.0_dp
- real(dp), parameter, public mo_mpr_constants::vgenuchten_sandtresh = 66.5_dp
- real(dp), parameter, public mo_mpr_constants::vgenuchtenn_c1 = 1.392_dp
- real(dp), parameter, public mo_mpr_constants::vgenuchtenn_c2 = 0.418_dp
- real(dp), parameter, public mo_mpr_constants::vgenuchtenn_c3 = -0.024_dp
- real(dp), parameter, public mo_mpr_constants::vgenuchtenn_c4 = 1.212_dp
- real(dp), parameter, public mo_mpr_constants::vgenuchtenn_c5 = -0.704_dp
- real(dp), parameter, public mo_mpr_constants::vgenuchtenn_c6 = -0.648_dp
- real(dp), parameter, public mo_mpr_constants::vgenuchtenn_c7 = 0.023_dp
- real(dp), parameter, public mo_mpr_constants::vgenuchtenn_c8 = 0.044_dp
- real(dp), parameter, public mo_mpr_constants::vgenuchtenn_c9 = 3.168_dp
- real(dp), parameter, public mo_mpr_constants::vgenuchtenn_c10 = -2.562_dp
- real(dp), parameter, public mo_mpr_constants::vgenuchtenn_c11 = 7.0E-9_dp
- real(dp), parameter, public mo_mpr_constants::vgenuchtenn_c12 = 4.004_dp
- real(dp), parameter, public mo_mpr_constants::vgenuchtenn_c13 = 3.750_dp
- real(dp), parameter, public mo_mpr_constants::vgenuchtenn_c14 = -0.016_dp
- real(dp), parameter, public mo_mpr_constants::vgenuchtenn_c15 = -4.197_dp
- real(dp), parameter, public mo_mpr_constants::vgenuchtenn_c16 = 0.013_dp
- real(dp), parameter, public mo_mpr_constants::vgenuchtenn_c17 = 0.076_dp
- real(dp), parameter, public mo_mpr_constants::vgenuchtenn_c18 = 0.276_dp
- real(dp), parameter, public mo_mpr_constants::ks_c = 10.0_dp
- real(dp), parameter, public mo_mpr_constants::pwp_c = 1.0_dp
- real(dp), parameter, public mo_mpr_constants::pwp_matpot_thetar = 15000.0_dp
- real(dp), parameter, public mo_mpr_constants::windmeasheight = 10.0_dp

  assumed meteorol. measurement hight for estimation of aeroResist and surfResist
- real(dp), parameter, public mo_mpr_constants::karman = 0.41_dp

  von karman constant
- real(dp), parameter, public mo_mpr_constants::lai_factor_surfresi = 0.3_dp

  LAI factor for bulk surface resistance formulation.
- real(dp), parameter, public mo_mpr_constants::lai_offset_surfresi = 1.2_dp

  LAI offset for bulk surface resistance formulation.
- real(dp), parameter, public mo_mpr_constants::max_surfresist = 250.0_dp

  maximum bulk surface resistance

18.48 mo_mpr_eval.f90 File Reference

Modules

- module mo_mpr_eval

  Runs MPR and writes to global effective parameters.

Functions/Subroutines

- subroutine, public mo_mpr_eval::mpr_eval (parameterset)

  Runs MPR and writes to global effective parameters.

18.49 mo_mpr_file.f90 File Reference

Modules

- module mo_mpr_file

  Provides file names and units for mRM.

Variables

- character(len=∗), parameter mo_mpr_file::version = '0.1'

  Current mHM model version.
- character(len=∗), parameter mo_mpr_file::version_date = 'Jun 2019'

  Time of current mHM model version release.
- character(len=∗), parameter mo_mpr_file::file_main = 'mpr_driver.f90'

  Driver file.
- character(len=∗), parameter mo_mpr_file::file_namelist_mpr = 'mpr.nml'

  Namelist file name.
- integer, parameter mo_mpr_file::unamelist_mpr = 80

  Unit for namelist.
- character(len=∗), parameter mo_mpr_file::file_namelist_mpr_param = 'mpr_parameter.nml'

  Parameter namelists file name.
- integer, parameter mo_mpr_file::unamelist_mpr_param = 31

  Unit for namelist.
- character(len=∗), parameter mo_mpr_file::file_soil_database = 'soil_classdefinition.txt'

  Soil database file (iFlag_soilDB = 0) = classical mHM format.
- character(len=∗), parameter mo_mpr_file::file_soil_database_1 = 'soil_classdefinition_iFlag_soilDB_1.txt'

  Soil database file (iFlag_soilDB = 1)
- integer, parameter mo_mpr_file::usoil_database = 52

  Unit for soil data base.
- character(len=∗), parameter mo_mpr_file::file_slope = 'slope.asc'

  slope input data file
- integer, parameter mo_mpr_file::uslope = 54

  Unit for slope input data file.
- character(len=∗), parameter mo_mpr_file::file_aspect = 'aspect.asc'

  aspect input data file
- integer, parameter mo_mpr_file::uaspect = 55

  Unit for aspect input data file.

- character(len=∗), parameter mo_mpr_file::file_hydrogeoclass = 'geology_class.asc'

  hydrogeological classes input data file
- integer, parameter mo_mpr_file::uhydrogeoclass = 58

  Unit for hydrogeological classes input data file.
- character(len=∗), parameter mo_mpr_file::file_soilclass = 'soil_class.asc'

  soil classes input data file
- integer, parameter mo_mpr_file::usoilclass = 59

  Unit for soil classes input data file.
- character(len=∗), parameter mo_mpr_file::file_laiclass = 'LAI_class.asc'

  LAI classes input data file.
- integer, parameter mo_mpr_file::ulaiclass = 60

  Unit for LAI input data file.
- character(len=∗), parameter mo_mpr_file::file_geolut = 'geology_classdefinition.txt'

  geological formation lookup table file
- integer, parameter mo_mpr_file::ugeolut = 64

  Unit for geological formation lookup table file.
- character(len=∗), parameter mo_mpr_file::file_lailut = 'LAI_classdefinition.txt'

  LAI classes lookup table file.
- integer, parameter mo_mpr_file::ulailut = 65

  Unit for LAI classes lookup table file.
- character(len=∗), parameter mo_mpr_file::file_meteo_header = 'header.txt'

  Input nCols and nRows of binary meteo and LAI files are in header file.
- integer, parameter mo_mpr_file::umeteo_header = 50

  Unit for meteo header file.
- character(len=∗), parameter mo_mpr_file::file_meteo_binary_end = '.bin'

  File ending of meteo files.
- integer, parameter mo_mpr_file::umeteo = 51

  Unit for meteo files.

18.50 mo_mpr_global_variables.f90 File Reference

Data Types

- type mo_mpr_global_variables::soiltype

Modules

- module mo_mpr_global_variables

  Global variables for mpr only.

Variables

- real(dp), public mo_mpr_global_variables::tillagedepth
- integer(i4), public mo_mpr_global_variables::nsoiltypes
- integer(i4), public mo_mpr_global_variables::iflag_soildb
- integer(i4), public mo_mpr_global_variables::nsoilhorizons_mhm
- real(dp), dimension(:), allocatable, public mo_mpr_global_variables::horizondepth_mhm
- type(soiltype), public mo_mpr_global_variables::soildb
- integer(i4), public mo_mpr_global_variables::ngeounits
- integer(i4), dimension(:), allocatable, public mo_mpr_global_variables::geounitlist

- integer(i4), dimension(:), allocatable, public mo_mpr_global_variables::geounitkar
- character(256), public mo_mpr_global_variables::inputformat_gridded_lai
- integer(i4), public mo_mpr_global_variables::timestep_lai_input
- integer(i4), public mo_mpr_global_variables::nlaiclass
- integer(i4), public mo_mpr_global_variables::nlai
- integer(i4), dimension(:), allocatable, public mo_mpr_global_variables::laiunitlist
- real(dp), dimension(:, :), allocatable, public mo_mpr_global_variables::lailut
- type(period), dimension(:), allocatable, public mo_mpr_global_variables::laiper
- real(dp), public mo_mpr_global_variables::fracsealed_cityarea
- real(dp), dimension(:), allocatable, public mo_mpr_global_variables::l0_slope_emp
- real(dp), dimension(:, :), allocatable, public mo_mpr_global_variables::l0_gridded_lai
- real(dp), dimension(:), allocatable, public mo_mpr_global_variables::l0_slope
- real(dp), dimension(:), allocatable, public mo_mpr_global_variables::l0_asp
- integer(i4), dimension(:, :), allocatable, public mo_mpr_global_variables::l0_soilid
- integer(i4), dimension(:), allocatable, public mo_mpr_global_variables::l0_geounit
- character(256), dimension(:), allocatable, public mo_mpr_global_variables::dirgridded_lai
- real(dp), dimension(:, :, :), allocatable, public mo_mpr_global_variables::l1_fsealed
- real(dp), dimension(:, :, :), allocatable, public mo_mpr_global_variables::l1_alpha
- real(dp), dimension(:, :, :), allocatable, public mo_mpr_global_variables::l1_degdayinc
- real(dp), dimension(:, :, :), allocatable, public mo_mpr_global_variables::l1_degdaymax
- real(dp), dimension(:, :, :), allocatable, public mo_mpr_global_variables::l1_degdaynopre
- real(dp), dimension(:, :, :), allocatable, public mo_mpr_global_variables::l1_degday
- real(dp), dimension(:, :, :), allocatable, public mo_mpr_global_variables::l1_karstloss
- real(dp), dimension(:, :, :), allocatable, public mo_mpr_global_variables::l1_fasp
- real(dp), dimension(:, :, :), allocatable, public mo_mpr_global_variables::l1_petlaicorfactor
- real(dp), dimension(:, :, :), allocatable, public mo_mpr_global_variables::l1_harsamcoeff
- real(dp), dimension(:, :, :), allocatable, public mo_mpr_global_variables::l1_prietayalpha
- real(dp), dimension(:, :, :), allocatable, public mo_mpr_global_variables::l1_aeroresist
- real(dp), dimension(:, :, :), allocatable, public mo_mpr_global_variables::l1_surfresist
- real(dp), dimension(:, :, :), allocatable, public mo_mpr_global_variables::l1_froots
- real(dp), dimension(:, :, :), allocatable, public mo_mpr_global_variables::l1_maxinter
- real(dp), dimension(:, :, :), allocatable, public mo_mpr_global_variables::l1_kfastflow
- real(dp), dimension(:, :, :), allocatable, public mo_mpr_global_variables::l1_kslowflow
- real(dp), dimension(:, :, :), allocatable, public mo_mpr_global_variables::l1_kbaseflow
- real(dp), dimension(:, :, :), allocatable, public mo_mpr_global_variables::l1_kperco
- real(dp), dimension(:, :, :), allocatable, public mo_mpr_global_variables::l1_soilmoistfc
- real(dp), dimension(:, :, :), allocatable, public mo_mpr_global_variables::l1_soilmoistsat
- real(dp), dimension(:, :, :), allocatable, public mo_mpr_global_variables::l1_soilmoistexp
- real(dp), dimension(:, :, :), allocatable, public mo_mpr_global_variables::l1_jarvis_thresh_c1
- real(dp), dimension(:, :, :), allocatable, public mo_mpr_global_variables::l1_tempthresh
- real(dp), dimension(:, :, :), allocatable, public mo_mpr_global_variables::l1_unsatthresh
- real(dp), dimension(:, :, :), allocatable, public mo_mpr_global_variables::l1_sealedthresh
- real(dp), dimension(:, :, :), allocatable, public mo_mpr_global_variables::l1_wiltingpoint

18.51 mo_mpr_pet.f90 File Reference

Modules

- module mo_mpr_pet

  TODO: add description.

Functions/Subroutines

- subroutine, public mo_mpr_pet::pet_correctbylai (param, nodata, LCOVER0, LAI0, mask0, cell_id0, upp_↩ row_L1, low_row_L1, lef_col_L1, rig_col_L1, nL0_in_L1, L1_petLAIcorFactor)

  estimate PET correction factor based on LAI at L1
- subroutine, public mo_mpr_pet::pet_correctbyasp (Id0, latitude_l0, Asp0, param, nodata, fAsp0)

  correction of PET
- subroutine, public mo_mpr_pet::priestley_taylor_alpha (LAI0, param, mask0, nodata, cell_id0, nL0_in_L1, Upp_row_L1, Low_row_L1, Lef_col_L1, Rig_col_L1, priestley_taylor_alpha1)

  Regionalization of priestley taylor alpha.
- subroutine, public mo_mpr_pet::bulksurface_resistance (LAI0, param, mask0, nodata, cell_id0, nL0_in_L1, Upp_row_L1, Low_row_L1, Lef_col_L1, Rig_col_L1, bulksurface_resistance1)

  Regionalization of bulk surface resistance.

18.52 mo_mpr_read_config.f90 File Reference

Modules

- module mo_mpr_read_config

  read mpr config

Functions/Subroutines

- subroutine, public mo_mpr_read_config::mpr_read_config (file_namelist, unamelist, file_namelist_param, unamelist_param)

  Read the general config of mpr.

18.53 mo_mpr_restart.f90 File Reference

Data Types

- interface mo_mpr_restart::unpack_field_and_write

  TODO: add description.

Modules

- module mo_mpr_restart

  reading and writing states, fluxes and configuration for restart of mHM.

Functions/Subroutines

- subroutine, public mo_mpr_restart::write_mpr_restart_files (OutPath)

  write restart files for each basin
- subroutine, public mo_mpr_restart::write_eff_params (mask1, s1, e1, rows1, cols1, soil1, lcscenes, lais, nc)

  TODO: add description.
- subroutine mo_mpr_restart::unpack_field_and_write_1d_i4 (nc, var_name, var_dims, fill_value, data, mask, var_long_name)
- subroutine mo_mpr_restart::unpack_field_and_write_1d_dp (nc, var_name, var_dims, fill_value, data, mask, var_long_name)

- subroutine mo_mpr_restart::unpack_field_and_write_2d_dp (nc, var_name, var_dims, fill_value, data, mask, var_long_name)
- subroutine mo_mpr_restart::unpack_field_and_write_3d_dp (nc, var_name, var_dims, fill_value, data, mask, var_long_name)

18.54 mo_mpr_runoff.f90 File Reference

Modules

- module mo_mpr_runoff

  multiscale parameter regionalization for runoff generation

Functions/Subroutines

- subroutine, public mo_mpr_runoff::mpr_runoff (LCOVER0, mask0, SMs_FC0, slope_emp0, KsVar_H0, param, cell_id0, upp_row_L1, low_row_L1, lef_col_L1, rig_col_L1, nL0_in_L1, L1_HL1, L1_K0, L1_K1, L1↩_alpha)

  multiscale parameter regionalization for runoff parameters

18.55 mo_mpr_smhorizons.f90 File Reference

Modules

- module mo_mpr_smhorizons

  setting up the soil moisture horizons

Functions/Subroutines

- subroutine, public mo_mpr_smhorizons::mpr_smhorizons (param, processMatrix, iFlag_soil, nHorizons_↩mHM, HorizonDepth, LCOVER0, soilID0, nHorizons, nTillHorizons, thetaS_till, thetaFC_till, thetaPW_till, thetaS, thetaFC, thetaPW, Wd, Db, DbM, RZdepth, mask0, cell_id0, upp_row_L1, low_row_L1, lef_col_↩L1, rig_col_L1, nL0_in_L1, L1_beta, L1_SMs, L1_FC, L1_PW, L1_fRoots)

  upscale soil moisture horizons

18.56 mo_mpr_soilmoist.f90 File Reference

Modules

- module mo_mpr_soilmoist

  Multiscale parameter regionalization (MPR) for soil moisture.

Functions/Subroutines

- subroutine, public mo_mpr_soilmoist::mpr_sm (param, is_present, nHorizons, nTillHorizons, sand, clay, DbM, ID0, soilId0, LCover0, thetaS_till, thetaFC_till, thetaPW_till, thetaS, thetaFC, thetaPW, Ks, Db, KsVar_H0, KsVar_V0, SMs_FC0)

  multiscale parameter regionalization for soil moisture
- elemental pure subroutine mo_mpr_soilmoist::pwp (Genu_Mual_n, Genu_Mual_alpha, thetaS, thetaPWP)

  Permanent Wilting point.

- elemental pure subroutine mo_mpr_soilmoist::field_cap (thetaFC, Ks, thetaS, Genu_Mual_n)

  calculates the field capacity
- subroutine mo_mpr_soilmoist::genuchten (thetaS, Genu_Mual_n, Genu_Mual_alpha, param, sand, clay, Db)

  calculates the Genuchten shape parameter
- subroutine mo_mpr_soilmoist::hydro_cond (KS, param, sand, clay)

  calculates the hydraulic conductivity Ks

18.57 mo_mpr_startup.f90 File Reference

Modules

- module mo_mpr_startup

  Startup procedures for mHM.

Functions/Subroutines

- subroutine, public mo_mpr_startup::mpr_initialize

  Initialize main mHM variables.
- subroutine mo_mpr_startup::l0_check_input (iBasin)

  Check for errors in L0 input data.
- subroutine mo_mpr_startup::l0_variable_init (iBasin)

  level 0 variable initialization
- subroutine, public mo_mpr_startup::init_eff_params (ncells1)

  Allocation of space for mHM related L1 and L11 variables.

18.58 mo_mrm_constants.f90 File Reference

Modules

- module mo_mrm_constants

  Provides mRM specific constants.

Variables

- integer(i4), parameter, public mo_mrm_constants::noutflxstate = 1_i4
- integer(i4), parameter, public mo_mrm_constants::nroutingstates = 2
- integer(i4), parameter, public mo_mrm_constants::maxnogauges = 50_i4
- real(dp), parameter, public mo_mrm_constants::rout_space_weight = 0._dp
- real(dp), parameter, public mo_mrm_constants::deltah = 5.000_dp
- real(dp), dimension(19), parameter mo_mrm_constants::given_ts = (/ 60._dp, 120._dp, 180._dp, 240._↩ dp, 300._dp, 360._dp, 600._dp, 720._dp, 900._dp, 1200._dp, 1800._dp, 3600._dp, 7200._dp, 10800._dp, 14400._dp, 21600._dp, 28800._dp, 43200._dp, 86400._dp/)

18.59 mo_mrm_eval.f90 File Reference

Modules

- module mo_mrm_eval

  Runs mrm with a specific parameter set and returns required variables, e.g. runoff.

Functions/Subroutines

- subroutine, public mo_mrm_eval::mrm_eval (parameterset, runoff, sm_opti, basin_avg_tws, neutrons_opti, et_opti)

  Runs mrm with a specific parameter set and returns required variables, e.g. runoff.

18.60 mo_mrm_file.f90 File Reference

Modules

- module mo_mrm_file

  Provides file names and units for mRM.

Variables

- character(len=∗), parameter mo_mrm_file::version = '1.0'

  Current mHM model version.
- character(len=∗), parameter mo_mrm_file::version_date = 'May 2019'

  Time of current mHM model version release.
- character(len=∗), parameter mo_mrm_file::file_main = 'mrm_driver.f90'

  Driver file.
- character(len=∗), parameter mo_mrm_file::file_namelist_mrm = 'mrm.nml'

  Namelist file name.
- integer, parameter mo_mrm_file::unamelist_mrm = 40

  Unit for namelist.
- character(len=∗), parameter mo_mrm_file::file_namelist_param_mrm = 'mrm_parameter.nml'

  Parameter namelists file name.
- integer, parameter mo_mrm_file::unamelist_param_mrm = 41

  Unit for namelist.
- character(len=∗), parameter mo_mrm_file::file_facc = 'facc.asc'
- integer, parameter mo_mrm_file::ufacc = 56

  Unit for flow accumulation input data file.
- character(len=∗), parameter mo_mrm_file::file_fdir = 'fdir.asc'

  flow direction input data file
- integer, parameter mo_mrm_file::ufdir = 57

  Unit for flow direction input data file.
- character(len= ∗), parameter mo_mrm_file::file_slope = 'slope.asc'

  flow direction input data file
- integer, parameter mo_mrm_file::uslope = 59

  Unit for flow direction input data file.
- character(len=∗), parameter mo_mrm_file::file_gaugeloc = 'idgauges.asc'

  gauge location input data file
- integer, parameter mo_mrm_file::ugaugeloc = 62

  Unit for gauge location input data file.
- integer, parameter mo_mrm_file::udischarge = 66

  unit for discharge time series
- character(len=∗), parameter mo_mrm_file::file_defoutput = 'mrm_outputs.nml'

  file defining mRM's outputs
- integer, parameter mo_mrm_file::udefoutput = 67

  Unit for file defining mRM's outputs.

- character(len=∗), parameter mo_mrm_file::file_config = 'ConfigFile.log'

  file defining mHM's outputs
- integer, parameter mo_mrm_file::uconfig = 68

  Unit for file defining mHM's outputs.
- character(len=∗), parameter mo_mrm_file::file_daily_discharge = 'daily_discharge.out'

  file defining optimazation outputs
- integer, parameter mo_mrm_file::udaily_discharge = 74

  Unit for file optimazation outputs.
- character(len=∗), parameter mo_mrm_file::ncfile_discharge = 'discharge.nc'

  file defining optimazation outputs
- character(len=∗), parameter mo_mrm_file::file_mrm_output = 'mRM_Fluxes_States.nc'

  file containing mrm output
- character(len=∗), parameter mo_mrm_file::file_gw_output = 'mRM_gw_Fluxes_States.nc'

  file containing mrm output for groundwater coupling

18.61 mo_mrm_global_variables.f90 File Reference

Data Types

- type mo_mrm_global_variables::gaugingstation
- type mo_mrm_global_variables::basininfo_mrm

Modules

- module mo_mrm_global_variables

  Global variables for mRM only.

Variables

- logical mo_mrm_global_variables::is_start
- integer(i4) mo_mrm_global_variables::timestep_model_outputs_mrm
- logical, dimension(noutflxstate) mo_mrm_global_variables::outputflxstate_mrm
- character(256), dimension(:), allocatable, public mo_mrm_global_variables::dirgauges
- character(256), dimension(:), allocatable, public mo_mrm_global_variables::dirtotalrunoff
- character(256), public mo_mrm_global_variables::filenametotalrunoff
- character(256), public mo_mrm_global_variables::varnametotalrunoff
- character(256), dimension(:), allocatable, public mo_mrm_global_variables::dirbankfullrunoff
- integer(i4), public mo_mrm_global_variables::ntstepday
- type(grid), dimension(:), allocatable, target, public mo_mrm_global_variables::level11
- type(gridremapper), dimension(:), allocatable, public mo_mrm_global_variables::l0_l11_remap
- type(gridremapper), dimension(:), allocatable, public mo_mrm_global_variables::l1_l11_remap
- real(dp), dimension(:, :), allocatable, public mo_mrm_global_variables::mrm_runoff
- integer(i4), public mo_mrm_global_variables::ngaugestotal
- integer(i4), public mo_mrm_global_variables::ninflowgaugestotal
- integer(i4), public mo_mrm_global_variables::nmeasperday
- type(gaugingstation), public mo_mrm_global_variables::gauge
- type(gaugingstation), public mo_mrm_global_variables::inflowgauge
- type(basininfo_mrm), dimension(:), allocatable, target, public mo_mrm_global_variables::basin_mrm
- integer(i4), dimension(:), allocatable, public mo_mrm_global_variables::l0_gaugeloc
- integer(i4), dimension(:), allocatable, public mo_mrm_global_variables::l0_inflowgaugeloc
- integer(i4), dimension(:), allocatable, public mo_mrm_global_variables::l0_facc

- integer(i4), dimension(:), allocatable, public mo_mrm_global_variables::l0_fdir
- integer(i4), dimension(:), allocatable, public mo_mrm_global_variables::l0_drasc
- integer(i4), dimension(:), allocatable, public mo_mrm_global_variables::l0_dracell
- integer(i4), dimension(:), allocatable, public mo_mrm_global_variables::l0_streamnet
- integer(i4), dimension(:), allocatable, public mo_mrm_global_variables::l0_floodplain
- integer(i4), dimension(:), allocatable, public mo_mrm_global_variables::l0_noutlet
- real(dp), dimension(:), allocatable, public mo_mrm_global_variables::l0_celerity
- integer(i4), dimension(:), allocatable, public mo_mrm_global_variables::l11_l1_id
- real(dp), dimension(:, :), allocatable, public mo_mrm_global_variables::l1_total_runoff_in
- integer(i4), dimension(:,:), allocatable, public mo_mrm_global_variables::l11_cellcoor
- integer(i4), dimension(:), allocatable, public mo_mrm_global_variables::l1_l11_id
- real(dp), dimension(:), allocatable, public mo_mrm_global_variables::l11_areacell
- real(dp), dimension(:), allocatable, public mo_mrm_global_variables::l11_facc
- integer(i4), dimension(:), allocatable, public mo_mrm_global_variables::l11_fdir
- integer(i4), dimension(:), allocatable, public mo_mrm_global_variables::l11_noutlets
- real(dp), dimension(:), allocatable, public mo_mrm_global_variables::l11_celerity
- real(dp), dimension(:), allocatable, public mo_mrm_global_variables::l11_meandering
- real(dp), dimension(:), allocatable, public mo_mrm_global_variables::l11_linkin_facc
- integer(i4), dimension(:), allocatable, public mo_mrm_global_variables::l11_rowout
- integer(i4), dimension(:), allocatable, public mo_mrm_global_variables::l11_colout
- real(dp), dimension(:), allocatable, public mo_mrm_global_variables::l11_qmod
- real(dp), dimension(:), allocatable, public mo_mrm_global_variables::l11_qout
- real(dp), dimension(:, :), allocatable, public mo_mrm_global_variables::l11_qtin
- real(dp), dimension(:, :), allocatable, public mo_mrm_global_variables::l11_qtr
- integer(i4), dimension(:), allocatable, public mo_mrm_global_variables::l11_fromn
- integer(i4), dimension(:), allocatable, public mo_mrm_global_variables::l11_ton
- integer(i4), dimension(:), allocatable, public mo_mrm_global_variables::l11_netperm
- integer(i4), dimension(:), allocatable, public mo_mrm_global_variables::l11_frow
- integer(i4), dimension(:), allocatable, public mo_mrm_global_variables::l11_fcol
- integer(i4), dimension(:), allocatable, public mo_mrm_global_variables::l11_trow
- integer(i4), dimension(:), allocatable, public mo_mrm_global_variables::l11_tcol
- integer(i4), dimension(:), allocatable, public mo_mrm_global_variables::l11_rorder
- integer(i4), dimension(:), allocatable, public mo_mrm_global_variables::l11_label
- logical, dimension(:), allocatable, public mo_mrm_global_variables::l11_sink
- real(dp), dimension(:), allocatable, public mo_mrm_global_variables::l11_length
- real(dp), dimension(:), allocatable, target, public mo_mrm_global_variables::l11_afloodplain
- real(dp), dimension(:), allocatable, public mo_mrm_global_variables::l11_slope
- real(dp), dimension(:, :), allocatable, public mo_mrm_global_variables::l11_nlinkfracfpimp
- real(dp), dimension(:), allocatable, public mo_mrm_global_variables::l11_k
- real(dp), dimension(:), allocatable, public mo_mrm_global_variables::l11_xi
- real(dp), dimension(:), allocatable, public mo_mrm_global_variables::l11_tsrout
- real(dp), dimension(:), allocatable, public mo_mrm_global_variables::l11_c1
- real(dp), dimension(:), allocatable, public mo_mrm_global_variables::l11_c2
- logical mo_mrm_global_variables::gw_coupling
- real(dp), dimension(:), allocatable, public mo_mrm_global_variables::l11_bankfull_runoff_in
- real(dp), dimension(:), allocatable, public mo_mrm_global_variables::l0_channel_depth
- real(dp), dimension(:), allocatable, public mo_mrm_global_variables::l0_channel_elevation
- real(dp), dimension(:), allocatable, public mo_mrm_global_variables::l0_river_head_mon_sum
- real(dp), dimension(:), allocatable, public mo_mrm_global_variables::l0_slope

18.62 mo_mrm_init.f90 File Reference

Modules

- module mo_mrm_init

  Wrapper for initializing Routing.

Functions/Subroutines

- subroutine, public mo_mrm_init::mrm_init (file_namelist, unamelist, file_namelist_param, unamelist_param)

  Initialize all mRM variables at all levels (i.e., L0, L1, and L11).
- subroutine mo_mrm_init::print_startup_message (file_namelist, file_namelist_param)

  TODO: add description.
- subroutine mo_mrm_init::config_output

  TODO: add description.
- subroutine, public mo_mrm_init::variables_default_init_routing

  Default initalization mRM related L11 variables.
- subroutine mo_mrm_init::l0_check_input_routing (L0Basin_iBasin)

  TODO: add description.
- subroutine mo_mrm_init::variables_alloc_routing (iBasin)

  TODO: add description.

18.63 mo_mrm_mpr.f90 File Reference

Modules

- module mo_mrm_mpr

  Perform Multiscale Parameter Regionalization on Routing Parameters.

Functions/Subroutines

- subroutine, public mo_mrm_mpr::reg_rout (param, length, slope, fFPimp, TS, C1, C2)

  Regionalized routing.
- subroutine, public mo_mrm_mpr::mrm_init_param (iBasin, param)

  TODO: add description.
- subroutine, public mo_mrm_mpr::mrm_update_param (iBasin, param)

  TODO: add description.

18.64 mo_mrm_net_startup.f90 File Reference

Modules

- module mo_mrm_net_startup

  Startup drainage network for mHM.

Functions/Subroutines

- subroutine, public mo_mrm_net_startup::l11_l1_mapping (iBasin)

  TODO: add description.
- subroutine, public mo_mrm_net_startup::l11_flow_direction (iBasin)

  Determine the flow direction of the upscaled river network at level L11.
- subroutine, public mo_mrm_net_startup::l11_set_network_topology (iBasin)

  Set network topology.
- subroutine, public mo_mrm_net_startup::l11_routing_order (iBasin)

  Find routing order, headwater cells and sink.
- subroutine, public mo_mrm_net_startup::l11_link_location (iBasin)

  Estimate the LO (row,col) location for each routing link at level L11.
- subroutine, public mo_mrm_net_startup::l11_set_drain_outlet_gauges (iBasin)

  Draining cell identification and Set gauging node.
- subroutine, public mo_mrm_net_startup::l11_stream_features (iBasin)

  Stream features (stream network and floodplain)
- subroutine, public mo_mrm_net_startup::l11_fraction_sealed_floodplain (LCClassImp, do_init)

  Fraction of the flood plain with impervious cover.
- subroutine mo_mrm_net_startup::moveup (elev0, fDir0, fi, fj, ss, nn)

  TODO: add description.
- subroutine mo_mrm_net_startup::movedownonecell (fDir, iRow, jCol)

  TODO: add description.
- subroutine mo_mrm_net_startup::celllength (iBasin, fDir, iRow, jCol, iCoorSystem, length)

  TODO: add description.
- subroutine, public mo_mrm_net_startup::get_distance_two_lat_lon_points (lat1, long1, lat2, long2, distance_out)

  estimate distance in [m] between two points in a lat-lon
- subroutine, public mo_mrm_net_startup::l11_flow_accumulation (iBasin)

  Calculates L11 flow accumulation per grid cell.
- recursive subroutine calculate_l11_flow_accumulation (fDir, fAcc, ii, jj, nrow, ncol)
- subroutine, public mo_mrm_net_startup::l11_calc_celerity (iBasin, param)

  L11 celerity based on L0 elevation and L0 fAcc.

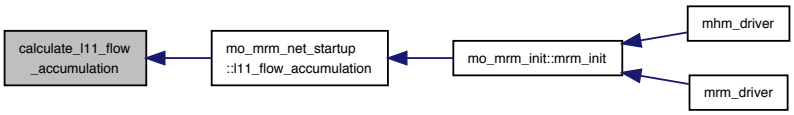
18.64.1 Function/Subroutine Documentation

18.64.1.1 calculate_l11_flow_accumulation()

```
recursive subroutine l11_flow_accumulation::calculate_l11_flow_accumulation (
            integer(i4), dimension(:,:), intent(in) fDir,
            real(dp), dimension(:,:), intent(inout) fAcc,
            integer(i4), intent(in) ii,
            integer(i4), intent(in) jj,
            integer(i4), intent(in) nrow,
            integer(i4), intent(in) ncol )
```

Referenced by mo_mrm_net_startup::l11_flow_accumulation().

Here is the caller graph for this function:



18.65 mo_mrm_objective_function_runoff.f90 File Reference

Modules

- module mo_mrm_objective_function_runoff

  Objective Functions for Optimization of mHM/mRM against runoff.

Functions/Subroutines

- real(dp) function, public mo_mrm_objective_function_runoff::single_objective_runoff (parameterset, eval, arg1, arg2, arg3)

  Wrapper for objective functions optimizing agains runoff.
- subroutine, public mo_mrm_objective_function_runoff::multi_objective_runoff (parameterset, eval, multi_objectives)

  Wrapper for multi-objective functions where at least one is regarding runoff.
- real(dp) function mo_mrm_objective_function_runoff::loglikelihood_stddev (parameterset, eval, stddev, stddev_new, likeli_new)

  Logarithmic likelihood function with removed linear trend and Lag(1)-autocorrelation.
- real(dp) function mo_mrm_objective_function_runoff::loglikelihood_evin2013_2 (parameterset, eval, regularize)

  Logarithmised likelihood with linear error model and lag(1)-autocorrelation of the relative errors.
- real(dp) function mo_mrm_objective_function_runoff::parameter_regularization (paraset, prior, bounds, mask)

  TODO: add description.
- real(dp) function mo_mrm_objective_function_runoff::loglikelihood_trend_no_autocorr (parameterset, eval, stddev_old, stddev_new, likeli_new)

  Logarithmic likelihood function with linear trend removed.
- real(dp) function mo_mrm_objective_function_runoff::objective_lnnse (parameterset, eval)

  Objective function of logarithmic NSE.
- real(dp) function mo_mrm_objective_function_runoff::objective_sse (parameterset, eval)

  Objective function of SSE.
- real(dp) function mo_mrm_objective_function_runoff::objective_nse (parameterset, eval)

  Objective function of NSE.
- real(dp) function mo_mrm_objective_function_runoff::objective_equal_nse_lnnse (parameterset, eval)

  Objective function equally weighting NSE and lnNSE.
- real(dp) function, dimension(2) mo_mrm_objective_function_runoff::multi_objective_nse_lnnse (parameterset, eval)

  Multi-objective function with NSE and lnNSE.
- real(dp) function, dimension(2) mo_mrm_objective_function_runoff::multi_objective_lnnse_highflow_lnnse_lowflow (parameterset, eval)

Multi-objective function with NSE and lnNSE.

- real(dp) function, dimension(2) mo_mrm_objective_function_runoff::multi_objective_lnnse_highflow_lnnse_lowflow_2 (parameterset, eval)

  Multi-objective function with NSE and lnNSE.
- real(dp) function, dimension(2) mo_mrm_objective_function_runoff::multi_objective_ae_fdc_lsv_nse_djf (parameterset, eval)

  Multi-objective function with absolute error of Flow Duration Curves low-segment volume and nse of DJF's discharge.
- real(dp) function mo_mrm_objective_function_runoff::objective_power6_nse_lnnse (parameterset, eval)

  Objective function of combined NSE and lnNSE with power of 5 i.e. the p-norm with p=5.
- real(dp) function mo_mrm_objective_function_runoff::objective_kge (parameterset, eval)

  Objective function of KGE.
- real(dp) function mo_mrm_objective_function_runoff::objective_multiple_gauges_kge_power6 (parameterset, eval)

  combined objective function based on KGE raised to the power 6
- real(dp) function mo_mrm_objective_function_runoff::objective_weighted_nse (parameterset, eval)

  Objective function of weighted NSE.
- subroutine, public mo_mrm_objective_function_runoff::extract_runoff (gaugeId, runoff, runoff_agg, runoff_obs, runoff_obs_mask)

  extracts runoff data from global variables

18.66 mo_mrm_read_config.f90 File Reference

Modules

- module mo_mrm_read_config

  read mRM config

Functions/Subroutines

- subroutine, public mo_mrm_read_config::mrm_read_config (file_namelist, unamelist, file_namelist_param, unamelist_param, do_message, readLatLon)

  Read the general config of mRM.
- subroutine mo_mrm_read_config::read_mrm_routing_params (processCase, file_namelist_param, unamelist_param)

  TODO: add description.

18.67 mo_mrm_read_data.f90 File Reference

Modules

- module mo_mrm_read_data

  This module contains all routines to read mRM data from file.

Functions/Subroutines

- subroutine, public mo_mrm_read_data::mrm_read_l0_data (do_reinit, do_readlatlon, do_readlcover)

  read L0 data from file
- subroutine, public mo_mrm_read_data::mrm_read_discharge

  Read discharge timeseries from file.

- subroutine, public mo_mrm_read_data::mrm_read_total_runoff (iBasin)

  read simulated runoff that is to be routed
- subroutine, public mo_mrm_read_data::mrm_read_bankfull_runoff (iBasin)
- subroutine mo_mrm_read_data::rotate_fdir_variable (x)

  TODO: add description.

18.68 mo_mrm_restart.f90 File Reference

Modules

- module mo_mrm_restart

  Restart routines.

Functions/Subroutines

- subroutine, public mo_mrm_restart::mrm_write_restart (iBasin, OutPath)

  write routing states and configuration
- subroutine, public mo_mrm_restart::mrm_read_restart_states (iBasin, InPath)

  read routing states
- subroutine, public mo_mrm_restart::mrm_read_restart_config (iBasin, InPath)

  reads Level 11 configuration from a restart directory

18.69 mo_mrm_river_head.f90 File Reference

Modules

- module mo_mrm_river_head

Functions/Subroutines

- subroutine, public mo_mrm_river_head::init_masked_zeros_l0 (iBasin, data)
- subroutine, private mo_mrm_river_head::reset_sum (iBasin, data)
- subroutine, public mo_mrm_river_head::calc_channel_elevation ()
- subroutine, public mo_mrm_river_head::calc_river_head (iBasin, L11_Qmod, river_head)
- real(dp) function mo_mrm_river_head::calc_slope (iBasin, elev0, fDir0, i, j)
- subroutine, public mo_mrm_river_head::avg_and_write_timestep (iBasin, timestep, data)
- subroutine, public mo_mrm_river_head::create_output (iBasin, OutPath)

Variables

- type(ncvariable), dimension(:), allocatable mo_mrm_river_head::nc_time
- type(ncvariable), dimension(:), allocatable mo_mrm_river_head::nc_riverhead
- integer(i4), dimension(:), allocatable mo_mrm_river_head::time_counter
- integer(i4), dimension(:), allocatable mo_mrm_river_head::sum_counter

18.70 mo_mrm_routing.f90 File Reference

Modules

- module mo_mrm_routing

  Performs runoff routing for mHM at level L11.

Functions/Subroutines

- subroutine, public mo_mrm_routing::mrm_routing (read_states, processCase, global_routing_param, L1_total_runoff, L1_areaCell, L1_L11_Id, L11_areaCell, L11_L1_Id, L11_netPerm, L11_fromN, L11_toN, L11_nOutlets, timestep, tsRoutFactor, nNodes, nInflowGauges, InflowGaugeIndexList, InflowGaugeHeadwater, InflowGaugeNodeList, InflowDischarge, nGauges, gaugeIndexList, gaugeNodeList, map_flag, L11_length, L11_slope, L11_FracFPimp, L11_C1, L11_C2, L11_qOut, L11_qTIN, L11_qTR, L11_qMod, GaugeDischarge)

  route water given runoff
- subroutine mo_mrm_routing::l11_runoff_acc (qAll, efecArea, L1_L11_Id, L11_areaCell, L11_L1_Id, TS, map_flag, qAcc)

  total runoff accumulation at L11.
- subroutine mo_mrm_routing::add_inflow (nInflowGauges, InflowIndexList, InflowHeadwater, InflowNodeList, QInflow, qOut)

  Adds inflow discharge to the runoff produced at the cell where the inflow is occurring.
- subroutine mo_mrm_routing::l11_routing (nNodes, nLinks, netPerm, netLink_fromN, netLink_toN, netLink_C1, netLink_C2, netNode_qOUT, nInflowGauges, InflowHeadwater, InflowNodeList, netNode_qTIN, netNode_qTR, netNode_Qmod)

  Performs runoff routing for mHM at L11 upscaled network (Routing Network).

18.71 mo_mrm_signatures.f90 File Reference

Modules

- module mo_mrm_signatures

  Module with calculations for several hydrological signatures.

Functions/Subroutines

- real(dp) function, dimension(size(lags, 1)), public mo_mrm_signatures::autocorrelation (data, lags, mask)

  Autocorrelation of a given data series.
- real(dp) function, dimension(size(quantiles, 1)), public mo_mrm_signatures::flowdurationcurve (data, quantiles, mask, concavity_index, mid_segment_slope, mhigh_segment_volume, high_segment_volume, low_segment_volume)

  Flow duration curves.
- subroutine, public mo_mrm_signatures::limb_densities (data, mask, RLD, DLD)

  Calculates limb densities.
- real(dp) function mo_mrm_signatures::maximummonthlyflow (data, mask, yr_start, mo_start, dy_start)

  Maximum of average flows per months.
- subroutine, public mo_mrm_signatures::moments (data, mask, mean_data, stddev_data, median_data, max_data, mean_log, stddev_log, median_log, max_log)

  Moments of data and log-transformed data, e.g. mean and standard deviation.
- real(dp) function, dimension(size(quantiles, 1)), public mo_mrm_signatures::peakdistribution (data, quantiles, mask, slope_peak_distribution)

Calculates the peak distribution.

- real(dp) function, public mo_mrm_signatures::runoffratio (data, basin_area, mask, precip_series, precip_↩ sum, log_data)

  Runoff ratio (accumulated daily discharge [mm/d] / accumulated daily precipitation [mm/d]).

- real(dp) function, public mo_mrm_signatures::zeroflowratio (data, mask)

  Ratio of zero values to total number of data points.

18.72 mo_mrm_write.f90 File Reference

Modules

- module mo_mrm_write

  write of discharge and restart files

Functions/Subroutines

- subroutine, public mo_mrm_write::mrm_write

  write discharge and restart files

- subroutine mo_mrm_write::write_configfile

  This modules writes the results of the configuration into an ASCII-file.

- subroutine mo_mrm_write::write_daily_obs_sim_discharge (Qobs, Qsim)

  Write a file for the daily observed and simulated discharge timeseries during the evaluation period for each gauging station.

- subroutine, public mo_mrm_write::mrm_write_output_fluxes (iBasin, nCells, timeStep_model_outputs, warmingDays, newTime, nTimeSteps, nTStepDay, tt, day, month, year, timestep, mask11, L11_qmod)

  write fluxes to netcdf output files

- subroutine, public mo_mrm_write::mrm_write_optifile (best_OF, best_paramSet, param_names)

  Write briefly final optimization results.

- subroutine, public mo_mrm_write::mrm_write_optinamelist (parameters, maskpara, parameters_name)

  Write final, optimized parameter set in a namelist format.

Variables

- integer(i4) mo_mrm_write::day_counter
- integer(i4) mo_mrm_write::month_counter
- integer(i4) mo_mrm_write::year_counter
- integer(i4) mo_mrm_write::average_counter
- type(outputdataset) mo_mrm_write::nc

18.73 mo_mrm_write_fluxes_states.f90 File Reference

Data Types

- interface mo_mrm_write_fluxes_states::outputvariable
- interface mo_mrm_write_fluxes_states::outputvariable
- interface mo_mrm_write_fluxes_states::outputdataset
- interface mo_mrm_write_fluxes_states::outputdataset

Modules

- module mo_mrm_write_fluxes_states

  Creates NetCDF output for different fluxes and state variables of mHM.

Functions/Subroutines

- type(outputvariable) function mo_mrm_write_fluxes_states::newoutputvariable (nc, name, dtype, dims, ncells, mask, avg)

  Initialize OutputVariable.
- subroutine mo_mrm_write_fluxes_states::updatevariable (self, data)

  Update OutputVariable.
- subroutine mo_mrm_write_fluxes_states::writevariabletimestep (self, timestep)

  Write timestep to file.
- type(outputdataset) function mo_mrm_write_fluxes_states::newoutputdataset (ibasin, mask, nCells)

  Initialize OutputDataset.
- subroutine mo_mrm_write_fluxes_states::updatedataset (self, sidx, eidx, L11_Qmod)

  Update all variables.
- subroutine mo_mrm_write_fluxes_states::writetimestep (self, timestep)

  Write all accumulated data.
- subroutine mo_mrm_write_fluxes_states::close (self)

  Close the file.
- type(ncdataset) function mo_mrm_write_fluxes_states::createoutputfile (ibasin)

  Create and initialize output file.
- subroutine mo_mrm_write_fluxes_states::writevariableattributes (var, long_name, unit)

  Write output variable attributes.

18.74 mo_multi_param_reg.f90 File Reference

Modules

- module mo_multi_param_reg

  Multiscale parameter regionalization (MPR).

Functions/Subroutines

- subroutine, public mo_multi_param_reg::mpr (mask0, geoUnit0, soilId0, Asp0, gridded_LAI0, LCover0, slope_emp0, y0, Id0, upper_bound1, lower_bound1, left_bound1, right_bound1, n_subcells1, fSealed1, alpha1, degDayInc1, degDayMax1, degDayNoPre1, fAsp1, HarSamCoeff1, PrieTayAlpha1, aeroResist1, surfResist1, fRoots1, kFastFlow1, kSlowFlow1, kBaseFlow1, kPerco1, karstLoss1, soilMoistFC1, soilMoistSat1, soilMoistExp1, jarvis_thresh_c1, tempThresh1, unsatThresh1, sealedThresh1, wiltingPoint1, maxInter1, petLAIcorFactor, parameterset)

  Regionalizing and Upscaling process parameters.
- subroutine mo_multi_param_reg::baseflow_param (param, geoUnit0, k2_0)

  baseflow recession parameter
- subroutine mo_multi_param_reg::snow_acc_melt_param (param, fForest1, fIperm1, fPerm1, tempThresh1, degDayNoPre1, degDayInc1, degDayMax1)

  Calculates the snow parameters.
- subroutine mo_multi_param_reg::iper_thres_runoff (param, sealedThresh1)

  sets the impervious layer threshold parameter for runoff generation

- subroutine mo_multi_param_reg::karstic_layer (param, geoUnit0, mask0, SMs_FC0, KsVar_V0, Id0, n_↩ subcells1, upper_bound1, lower_bound1, left_bound1, right_bound1, karstLoss1, L1_Kp)

  calculates the Karstic percolation loss
- subroutine, public mo_multi_param_reg::canopy_intercept_param (processMatrix, param, LAI0, n_subcells1, upper_bound1, lower_bound1, left_bound1, right_bound1, Id0, mask0, nodata, max_intercept1)

  estimate effective maximum interception capacity at L1
- subroutine mo_multi_param_reg::aerodynamical_resistance (LAI0, LCover0, param, mask0, Id0, n_↩ subcells1, upper_bound1, lower_bound1, left_bound1, right_bound1, aerodyn_resistance1)

  Regionalization of aerodynamic resistance.

18.75 mo_ncread.f90 File Reference

Data Types

- interface mo_ncread::get_ncvar

Modules

- module mo_ncread

Functions/Subroutines

- integer(i4) function, dimension(5), public mo_ncread::get_ncdim (Filename, Variable, PrintInfo, ndims)
- subroutine, public mo_ncread::get_ncdimatt (Filename, Variable, DimName, DimLen)
- subroutine, public mo_ncread::get_ncvaratt (FileName, VarName, AttName, AttValues, fid, dtype)
- subroutine mo_ncread::get_ncvar_0d_sp (Filename, VarName, Dat, fid)
- subroutine mo_ncread::get_ncvar_0d_dp (Filename, VarName, Dat, fid)
- subroutine mo_ncread::get_ncvar_1d_sp (Filename, VarName, Dat, start, a_count, fid)
- subroutine mo_ncread::get_ncvar_1d_dp (Filename, VarName, Dat, start, a_count, fid)
- subroutine mo_ncread::get_ncvar_2d_sp (Filename, VarName, Dat, start, a_count, fid)
- subroutine mo_ncread::get_ncvar_2d_dp (Filename, VarName, Dat, start, a_count, fid)
- subroutine mo_ncread::get_ncvar_3d_sp (Filename, VarName, Dat, start, a_count, fid)
- subroutine mo_ncread::get_ncvar_3d_dp (Filename, VarName, Dat, start, a_count, fid)
- subroutine mo_ncread::get_ncvar_4d_sp (Filename, VarName, Dat, start, a_count, fid)
- subroutine mo_ncread::get_ncvar_4d_dp (Filename, VarName, Dat, start, a_count, fid)
- subroutine mo_ncread::get_ncvar_5d_sp (Filename, VarName, Dat, start, a_count, fid)
- subroutine mo_ncread::get_ncvar_5d_dp (Filename, VarName, Dat, start, a_count, fid)
- subroutine mo_ncread::get_ncvar_0d_i4 (Filename, VarName, Dat, fid)
- subroutine mo_ncread::get_ncvar_1d_i4 (Filename, VarName, Dat, start, a_count, fid)
- subroutine mo_ncread::get_ncvar_2d_i4 (Filename, VarName, Dat, start, a_count, fid)
- subroutine mo_ncread::get_ncvar_3d_i4 (Filename, VarName, Dat, start, a_count, fid)
- subroutine mo_ncread::get_ncvar_4d_i4 (Filename, VarName, Dat, start, a_count, fid)
- subroutine mo_ncread::get_ncvar_5d_i4 (Filename, VarName, Dat, start, a_count, fid)
- subroutine mo_ncread::get_ncvar_0d_i1 (Filename, VarName, Dat, fid)
- subroutine mo_ncread::get_ncvar_1d_i1 (Filename, VarName, Dat, start, a_count, fid)
- subroutine mo_ncread::get_ncvar_2d_i1 (Filename, VarName, Dat, start, a_count, fid)
- subroutine mo_ncread::get_ncvar_3d_i1 (Filename, VarName, Dat, start, a_count, fid)
- subroutine mo_ncread::get_ncvar_4d_i1 (Filename, VarName, Dat, start, a_count, fid)
- subroutine mo_ncread::get_ncvar_5d_i1 (Filename, VarName, Dat, start, a_count, fid)
- integer(i4) function, public mo_ncread::ncopen (Fname)
- subroutine, public mo_ncread::ncclose (ncid)
- subroutine mo_ncread::get_info (Varname, ncid, varid, xtype, dl, Info, ndims)
- subroutine mo_ncread::check (status)

18.76 mo_ncwrite.f90 File Reference

Data Types

- type mo_ncwrite::dims
- type mo_ncwrite::attribute
- type mo_ncwrite::variable
- interface mo_ncwrite::dump_netcdf
- interface mo_ncwrite::var2nc

  Extended dump_netcdf for multiple variables.

Modules

- module mo_ncwrite

Functions/Subroutines

- subroutine, public mo_ncwrite::close_netcdf (ncId)
- subroutine, public mo_ncwrite::create_netcdf (Filename, ncid, lfs, netcdf4, deflate_level)
- subroutine mo_ncwrite::dump_netcdf_1d_sp (filename, arr, append, lfs, netcdf4, deflate_level)
- subroutine mo_ncwrite::dump_netcdf_2d_sp (filename, arr, append, lfs, netcdf4, deflate_level)
- subroutine mo_ncwrite::dump_netcdf_3d_sp (filename, arr, append, lfs, netcdf4, deflate_level)
- subroutine mo_ncwrite::dump_netcdf_4d_sp (filename, arr, append, lfs, netcdf4, deflate_level)
- subroutine mo_ncwrite::dump_netcdf_5d_sp (filename, arr, append, lfs, netcdf4, deflate_level)
- subroutine mo_ncwrite::dump_netcdf_1d_dp (filename, arr, append, lfs, netcdf4, deflate_level)
- subroutine mo_ncwrite::dump_netcdf_2d_dp (filename, arr, append, lfs, netcdf4, deflate_level)
- subroutine mo_ncwrite::dump_netcdf_3d_dp (filename, arr, append, lfs, netcdf4, deflate_level)
- subroutine mo_ncwrite::dump_netcdf_4d_dp (filename, arr, append, lfs, netcdf4, deflate_level)
- subroutine mo_ncwrite::dump_netcdf_5d_dp (filename, arr, append, lfs, netcdf4, deflate_level)
- subroutine mo_ncwrite::dump_netcdf_1d_i4 (filename, arr, append, lfs, netcdf4, deflate_level)
- subroutine mo_ncwrite::dump_netcdf_2d_i4 (filename, arr, append, lfs, netcdf4, deflate_level)
- subroutine mo_ncwrite::dump_netcdf_3d_i4 (filename, arr, append, lfs, netcdf4, deflate_level)
- subroutine mo_ncwrite::dump_netcdf_4d_i4 (filename, arr, append, lfs, netcdf4, deflate_level)
- subroutine mo_ncwrite::dump_netcdf_5d_i4 (filename, arr, append, lfs, netcdf4, deflate_level)
- subroutine mo_ncwrite::var2nc_1d_i4 (f_name, arr, dnames, v_name, dim_unlimited, long_name, units, missing_value, attributes, create, ncid, nrec)
- subroutine mo_ncwrite::var2nc_1d_sp (f_name, arr, dnames, v_name, dim_unlimited, long_name, units, missing_value, attributes, create, ncid, nrec)
- subroutine mo_ncwrite::var2nc_1d_dp (f_name, arr, dnames, v_name, dim_unlimited, long_name, units, missing_value, attributes, create, ncid, nrec)
- subroutine mo_ncwrite::var2nc_2d_i4 (f_name, arr, dnames, v_name, dim_unlimited, long_name, units, missing_value, attributes, create, ncid, nrec)
- subroutine mo_ncwrite::var2nc_2d_sp (f_name, arr, dnames, v_name, dim_unlimited, long_name, units, missing_value, attributes, create, ncid, nrec)
- subroutine mo_ncwrite::var2nc_2d_dp (f_name, arr, dnames, v_name, dim_unlimited, long_name, units, missing_value, attributes, create, ncid, nrec)
- subroutine mo_ncwrite::var2nc_3d_i4 (f_name, arr, dnames, v_name, dim_unlimited, long_name, units, missing_value, attributes, create, ncid, nrec)
- subroutine mo_ncwrite::var2nc_3d_sp (f_name, arr, dnames, v_name, dim_unlimited, long_name, units, missing_value, attributes, create, ncid, nrec)
- subroutine mo_ncwrite::var2nc_3d_dp (f_name, arr, dnames, v_name, dim_unlimited, long_name, units, missing_value, attributes, create, ncid, nrec)
- subroutine mo_ncwrite::var2nc_4d_i4 (f_name, arr, dnames, v_name, dim_unlimited, long_name, units, missing_value, attributes, create, ncid, nrec)

- subroutine mo_ncwrite::var2nc_4d_sp (f_name, arr, dnames, v_name, dim_unlimited, long_name, units, missing_value, attributes, create, ncid, nrec)
- subroutine mo_ncwrite::var2nc_4d_dp (f_name, arr, dnames, v_name, dim_unlimited, long_name, units, missing_value, attributes, create, ncid, nrec)
- subroutine mo_ncwrite::var2nc_5d_i4 (f_name, arr, dnames, v_name, dim_unlimited, long_name, units, missing_value, attributes, create, ncid, nrec)
- subroutine mo_ncwrite::var2nc_5d_sp (f_name, arr, dnames, v_name, dim_unlimited, long_name, units, missing_value, attributes, create, ncid, nrec)
- subroutine mo_ncwrite::var2nc_5d_dp (f_name, arr, dnames, v_name, dim_unlimited, long_name, units, missing_value, attributes, create, ncid, nrec)
- subroutine, public mo_ncwrite::write_dynamic_netcdf (ncId, irec)
- subroutine, public mo_ncwrite::write_static_netcdf (ncId)
- integer(i4) function mo_ncwrite::open_netcdf (f_name, create)
- subroutine mo_ncwrite::check (status)

Variables

- integer(i4), parameter, public mo_ncwrite::nmaxdim = 5
- integer(i4), parameter, public mo_ncwrite::nmaxatt = 20
- integer(i4), parameter, public mo_ncwrite::maxlen = 256
- integer(i4), parameter, public mo_ncwrite::ngatt = 20
- integer(i4), parameter, public mo_ncwrite::nattdim = 2
- integer(i4), public mo_ncwrite::nvars
- integer(i4), public mo_ncwrite::ndims
- type(dims), dimension(:), allocatable, public mo_ncwrite::dnc
- type(variable), dimension(:), allocatable, public mo_ncwrite::v
- type(attribute), dimension(ngatt), public mo_ncwrite::gatt

18.77 mo_netcdf.f90 File Reference

Data Types

- interface mo_netcdf::ncdataset

  Provides basic file modification functionality.
- interface mo_netcdf::ncdataset

  Provides basic file modification functionality.
- type mo_netcdf::ncdimension

  Provides the dimension access functionality.
- interface mo_netcdf::ncvariable

Modules

- module mo_netcdf

  NetCDF Fortran 90 interface wrapper.

Functions/Subroutines

- subroutine mo_netcdf::initncvariable (self, id, parent)
- subroutine mo_netcdf::initncdimension (self, id, parent)
- subroutine mo_netcdf::initncdataset (self, fname, mode)
- type(ncvariable) function mo_netcdf::newncvariable (id, parent)

- type(ncdimension) function mo_netcdf::newncdimension (id, parent)
- type(ncdataset) function mo_netcdf::newncdataset (fname, mode)
- subroutine mo_netcdf::close (self)
- integer(i4) function mo_netcdf::getnovariables (self)
- integer(i4) function, dimension(:), allocatable mo_netcdf::getvariableids (self)
- type(ncvariable) function, dimension(:), allocatable mo_netcdf::getvariables (self)
- character(len=256) function mo_netcdf::getdimensionname (self)
- integer(i4) function mo_netcdf::getdimensionlength (self)
- logical function mo_netcdf::isdatasetunlimited (self)
- type(ncdimension) function mo_netcdf::getunlimiteddimension (self)
- logical function mo_netcdf::equalncdimensions (dim1, dim2)
- logical function mo_netcdf::isunlimiteddimension (self)
- type(ncdimension) function mo_netcdf::setdimension (self, name, length)
- logical function mo_netcdf::hasvariable (self, name)
- logical function mo_netcdf::hasdimension (self, name)
- type(ncvariable) function mo_netcdf::setvariablewithids (self, name, dtype, dimensions, contiguous, chunksizes, deflate_level, shuffle, fletcher32, endianness, cache_size, cache_nelems, cache_preemption)
- type(ncvariable) function mo_netcdf::setvariablewithnames (self, name, dtype, dimensions, contiguous, chunksizes, deflate_level, shuffle, fletcher32, endianness, cache_size, cache_nelems, cache_preemption)
- type(ncvariable) function mo_netcdf::setvariablewithtypes (self, name, dtype, dimensions, contiguous, chunksizes, deflate_level, shuffle, fletcher32, endianness, cache_size, cache_nelems, cache_preemption)
- type(ncdimension) function mo_netcdf::getdimensionbyid (self, id)
- type(ncdimension) function mo_netcdf::getdimensionbyname (self, name)
- type(ncvariable) function mo_netcdf::getvariablebyname (self, name)
- character(len=256) function mo_netcdf::getvariablename (self)
- integer(i4) function mo_netcdf::getnodimensions (self)
- type(ncdimension) function, dimension(:), allocatable mo_netcdf::getvariabledimensions (self)
- integer(i4) function, dimension(:), allocatable mo_netcdf::getvariableshape (self)
- character(3) function mo_netcdf::getvariabledtype (self)
- logical function mo_netcdf::isunlimitedvariable (self)
- logical function mo_netcdf::hasattribute (self, name)
- subroutine mo_netcdf::setglobalattributechar (self, name, data)
- subroutine mo_netcdf::setglobalattributei8 (self, name, data)
- subroutine mo_netcdf::setglobalattributei16 (self, name, data)
- subroutine mo_netcdf::setglobalattributei32 (self, name, data)
- subroutine mo_netcdf::setglobalattributei64 (self, name, data)
- subroutine mo_netcdf::setglobalattributef32 (self, name, data)
- subroutine mo_netcdf::setglobalattributef64 (self, name, data)
- subroutine mo_netcdf::getglobalattributechar (self, name, avalue)
- subroutine mo_netcdf::getglobalattributei8 (self, name, avalue)
- subroutine mo_netcdf::getglobalattributei16 (self, name, avalue)
- subroutine mo_netcdf::getglobalattributei32 (self, name, avalue)
- subroutine mo_netcdf::getglobalattributei64 (self, name, avalue)
- subroutine mo_netcdf::getglobalattributef32 (self, name, avalue)
- subroutine mo_netcdf::getglobalattributef64 (self, name, avalue)
- subroutine mo_netcdf::setvariableattributechar (self, name, data)
- subroutine mo_netcdf::setvariableattributei8 (self, name, data)
- subroutine mo_netcdf::setvariableattributei16 (self, name, data)
- subroutine mo_netcdf::setvariableattributei32 (self, name, data)
- subroutine mo_netcdf::setvariableattributei64 (self, name, data)
- subroutine mo_netcdf::setvariableattributef32 (self, name, data)
- subroutine mo_netcdf::setvariableattributef64 (self, name, data)
- subroutine mo_netcdf::getvariableattributechar (self, name, avalue)
- subroutine mo_netcdf::getvariableattributei8 (self, name, avalue)
- subroutine mo_netcdf::getvariableattributei16 (self, name, avalue)

- subroutine mo_netcdf::getvariableattributei32 (self, name, avalue)
- subroutine mo_netcdf::getvariableattributei64 (self, name, avalue)
- subroutine mo_netcdf::getvariableattributef32 (self, name, avalue)
- subroutine mo_netcdf::getvariableattributef64 (self, name, avalue)
- subroutine mo_netcdf::setvariablefillvaluei8 (self, fvalue)
- subroutine mo_netcdf::setvariablefillvaluei16 (self, fvalue)
- subroutine mo_netcdf::setvariablefillvaluei32 (self, fvalue)
- subroutine mo_netcdf::setvariablefillvaluei64 (self, fvalue)
- subroutine mo_netcdf::setvariablefillvaluef32 (self, fvalue)
- subroutine mo_netcdf::setvariablefillvaluef64 (self, fvalue)
- subroutine mo_netcdf::getvariablefillvaluei8 (self, fvalue)
- subroutine mo_netcdf::getvariablefillvaluei16 (self, fvalue)
- subroutine mo_netcdf::getvariablefillvaluei32 (self, fvalue)
- subroutine mo_netcdf::getvariablefillvaluei64 (self, fvalue)
- subroutine mo_netcdf::getvariablefillvaluef32 (self, fvalue)
- subroutine mo_netcdf::getvariablefillvaluef64 (self, fvalue)
- subroutine mo_netcdf::setdatascalari8 (self, values, start)
- subroutine mo_netcdf::setdata1di8 (self, values, start, cnt, stride, map)
- subroutine mo_netcdf::setdata2di8 (self, values, start, cnt, stride, map)
- subroutine mo_netcdf::setdata3di8 (self, values, start, cnt, stride, map)
- subroutine mo_netcdf::setdata4di8 (self, values, start, cnt, stride, map)
- subroutine mo_netcdf::setdata5di8 (self, values, start, cnt, stride, map)
- subroutine mo_netcdf::setdatascalari16 (self, values, start)
- subroutine mo_netcdf::setdata1di16 (self, values, start, cnt, stride, map)
- subroutine mo_netcdf::setdata2di16 (self, values, start, cnt, stride, map)
- subroutine mo_netcdf::setdata3di16 (self, values, start, cnt, stride, map)
- subroutine mo_netcdf::setdata4di16 (self, values, start, cnt, stride, map)
- subroutine mo_netcdf::setdata5di16 (self, values, start, cnt, stride, map)
- subroutine mo_netcdf::setdatascalari32 (self, values, start)
- subroutine mo_netcdf::setdata1di32 (self, values, start, cnt, stride, map)
- subroutine mo_netcdf::setdata2di32 (self, values, start, cnt, stride, map)
- subroutine mo_netcdf::setdata3di32 (self, values, start, cnt, stride, map)
- subroutine mo_netcdf::setdata4di32 (self, values, start, cnt, stride, map)
- subroutine mo_netcdf::setdata5di32 (self, values, start, cnt, stride, map)
- subroutine mo_netcdf::setdatascalari64 (self, values, start)
- subroutine mo_netcdf::setdata1di64 (self, values, start, cnt, stride, map)
- subroutine mo_netcdf::setdata2di64 (self, values, start, cnt, stride, map)
- subroutine mo_netcdf::setdata3di64 (self, values, start, cnt, stride, map)
- subroutine mo_netcdf::setdata4di64 (self, values, start, cnt, stride, map)
- subroutine mo_netcdf::setdata5di64 (self, values, start, cnt, stride, map)
- subroutine mo_netcdf::setdatascalarf32 (self, values, start)
- subroutine mo_netcdf::setdata1df32 (self, values, start, cnt, stride, map)
- subroutine mo_netcdf::setdata2df32 (self, values, start, cnt, stride, map)
- subroutine mo_netcdf::setdata3df32 (self, values, start, cnt, stride, map)
- subroutine mo_netcdf::setdata4df32 (self, values, start, cnt, stride, map)
- subroutine mo_netcdf::setdata5df32 (self, values, start, cnt, stride, map)
- subroutine mo_netcdf::setdatascalarf64 (self, values, start)
- subroutine mo_netcdf::setdata1df64 (self, values, start, cnt, stride, map)
- subroutine mo_netcdf::setdata2df64 (self, values, start, cnt, stride, map)
- subroutine mo_netcdf::setdata3df64 (self, values, start, cnt, stride, map)
- subroutine mo_netcdf::setdata4df64 (self, values, start, cnt, stride, map)
- subroutine mo_netcdf::setdata5df64 (self, values, start, cnt, stride, map)
- subroutine mo_netcdf::getdatascalari8 (self, data, start, cnt, stride, map)
- subroutine mo_netcdf::getdata1di8 (self, data, start, cnt, stride, map)
- subroutine mo_netcdf::getdata2di8 (self, data, start, cnt, stride, map)

- subroutine mo_netcdf::getdata3di8 (self, data, start, cnt, stride, map)
- subroutine mo_netcdf::getdata4di8 (self, data, start, cnt, stride, map)
- subroutine mo_netcdf::getdata5di8 (self, data, start, cnt, stride, map)
- subroutine mo_netcdf::getdatascalari16 (self, data, start, cnt, stride, map)
- subroutine mo_netcdf::getdata1di16 (self, data, start, cnt, stride, map)
- subroutine mo_netcdf::getdata2di16 (self, data, start, cnt, stride, map)
- subroutine mo_netcdf::getdata3di16 (self, data, start, cnt, stride, map)
- subroutine mo_netcdf::getdata4di16 (self, data, start, cnt, stride, map)
- subroutine mo_netcdf::getdata5di16 (self, data, start, cnt, stride, map)
- subroutine mo_netcdf::getdatascalari32 (self, data, start, cnt, stride, map)
- subroutine mo_netcdf::getdata1di32 (self, data, start, cnt, stride, map)
- subroutine mo_netcdf::getdata2di32 (self, data, start, cnt, stride, map)
- subroutine mo_netcdf::getdata3di32 (self, data, start, cnt, stride, map)
- subroutine mo_netcdf::getdata4di32 (self, data, start, cnt, stride, map)
- subroutine mo_netcdf::getdata5di32 (self, data, start, cnt, stride, map)
- subroutine mo_netcdf::getdatascalari64 (self, data, start, cnt, stride, map)
- subroutine mo_netcdf::getdata1di64 (self, data, start, cnt, stride, map)
- subroutine mo_netcdf::getdata2di64 (self, data, start, cnt, stride, map)
- subroutine mo_netcdf::getdata3di64 (self, data, start, cnt, stride, map)
- subroutine mo_netcdf::getdata4di64 (self, data, start, cnt, stride, map)
- subroutine mo_netcdf::getdata5di64 (self, data, start, cnt, stride, map)
- subroutine mo_netcdf::getdatascalarf32 (self, data, start, cnt, stride, map)
- subroutine mo_netcdf::getdata1df32 (self, data, start, cnt, stride, map)
- subroutine mo_netcdf::getdata2df32 (self, data, start, cnt, stride, map)
- subroutine mo_netcdf::getdata3df32 (self, data, start, cnt, stride, map)
- subroutine mo_netcdf::getdata4df32 (self, data, start, cnt, stride, map)
- subroutine mo_netcdf::getdata5df32 (self, data, start, cnt, stride, map)
- subroutine mo_netcdf::getdatascalarf64 (self, data, start, cnt, stride, map)
- subroutine mo_netcdf::getdata1df64 (self, data, start, cnt, stride, map)
- subroutine mo_netcdf::getdata2df64 (self, data, start, cnt, stride, map)
- subroutine mo_netcdf::getdata3df64 (self, data, start, cnt, stride, map)
- subroutine mo_netcdf::getdata4df64 (self, data, start, cnt, stride, map)
- subroutine mo_netcdf::getdata5df64 (self, data, start, cnt, stride, map)
- integer(i4) function, dimension(datarank) mo_netcdf::getreaddatashape (var, datarank, instart, incnt, instride)
- integer(i4) function mo_netcdf::getdtypefromstring (dtype)
- character(3) function mo_netcdf::getdtypefrominteger (dtype)
- subroutine mo_netcdf::check (status, msg)

18.78 mo_neutrons.f90 File Reference

Modules

- module mo_neutrons

  Models to predict neutron intensities above soils.

Functions/Subroutines

- subroutine, public mo_neutrons::desiletsn0 (SoilMoisture, Horizons, N0, neutrons)

  Calculate neutrons from soil moisture in the first layer.
- subroutine, public mo_neutrons::cosmic (SoilMoisture, Horizons, params, neutron_integral_AFast, neutrons)

  Calculate neutrons from soil moisture in all layers.
- subroutine mo_neutrons::oldintegration (res, c)

TODO: add description.

- subroutine, public mo_neutrons::tabularintegralafast (integral, maxC)

  Save approximation data for A_fast.
- subroutine mo_neutrons::approx_mon_int (res, f, c, xmin, xmax, eps, steps, fxmin, fxmax)

  TODO: add description.
- recursive subroutine mo_neutrons::approx_mon_int_steps (res, f, c, xmin, xmax, eps, steps, fxmin, fxmax)

  TODO: add description.
- recursive subroutine mo_neutrons::approx_mon_int_eps (res, f, c, xmin, xmax, eps, fxmin, fxmax)

  TODO: add description.
- subroutine mo_neutrons::lookupintegral (res, integral, c)

  TODO: add description.
- real(dp) function mo_neutrons::intgrandfast (c, phi)

  TODO: add description.

18.79 mo_nml.f90 File Reference

Modules

- module mo_nml

  Deal with namelist files.

Functions/Subroutines

- subroutine, public mo_nml::open_nml (file, unit, quiet)

  Open a namelist file.
- subroutine, public mo_nml::close_nml (unit)

  Close a namelist file.
- subroutine, public mo_nml::position_nml (name, unit, status, first)

  Position a namlist file.

Variables

- integer(i4), parameter, public mo_nml::positioned = 0

  Information: file pointer set to namelist group.
- integer(i4), parameter, public mo_nml::missing = 1

  Error: namelist group is missing.
- integer(i4), parameter, public mo_nml::length_error = 2

  Error: namelist group name too long.
- integer(i4), parameter, public mo_nml::read_error = 3

  Error occured during read of namelist file.
- integer, save, public mo_nml::nunitnml = -1

18.80 mo_objective_function.f90 File Reference

Modules

- module mo_objective_function

  Objective Functions for Optimization of mHM.

Functions/Subroutines

- real(dp) function, public mo_objective_function::objective (parameterset, eval, arg1, arg2, arg3)

  Wrapper for objective functions.
- real(dp) function mo_objective_function::objective_sm_kge_catchment_avg (parameterset, eval)

  Objective function for soil moisture.
- real(dp) function mo_objective_function::objective_sm_corr (parameterset, eval)

  Objective function for soil moisture.
- real(dp) function mo_objective_function::objective_sm_pd (parameterset, eval)

  Objective function for soil moisture.
- real(dp) function mo_objective_function::objective_sm_sse_standard_score (parameterset, eval)

  Objective function for soil moisture.
- real(dp) function mo_objective_function::objective_kge_q_rmse_tws (parameterset, eval)

  Objective function of KGE for runoff and RMSE for basin_avg TWS (standarized scores)
- real(dp) function mo_objective_function::objective_neutrons_kge_catchment_avg (parameterset, eval)

  Objective function for neutrons.
- real(dp) function mo_objective_function::objective_et_kge_catchment_avg (parameterset, eval)

  Objective function for evpotranspirstion (et).
- real(dp) function mo_objective_function::objective_kge_q_sm_corr (parameterset, eval)

  Objective function of KGE for runoff and correlation for SM.
- real(dp) function mo_objective_function::objective_kge_q_et (parameterset, eval)

  Objective function of KGE for runoff and KGE for ET.
- real(dp) function mo_objective_function::objective_kge_q_rmse_et (parameterset, eval)

  Objective function of KGE for runoff and RMSE for basin_avg ET (standarized scores)
- subroutine mo_objective_function::extract_basin_avg_tws (basinId, tws, tws_sim, tws_obs, tws_obs_mask)

  extracts basin average tws data from global variables

18.81 mo_optimization.f90 File Reference

Modules

- module mo_optimization

  Wrapper subroutine for optimization against runoff and sm.

Functions/Subroutines

- subroutine, public mo_optimization::optimization (eval, objective, dirConfigOut, funcBest, maskpara)

  Wrapper for optimization.

18.82 mo_optimization_utils.f90 File Reference

Data Types

- interface mo_optimization_utils::eval_interface
- interface mo_optimization_utils::objective_interface

Modules

- module mo_optimization_utils

18.83 mo_orderpack.f90 File Reference

Data Types

- interface mo_orderpack::sort

  Unconditional ranking.
- interface mo_orderpack::sort_index
- interface mo_orderpack::ctrper
- interface mo_orderpack::fndnth
- interface mo_orderpack::indmed
- interface mo_orderpack::indnth
- interface mo_orderpack::inspar
- interface mo_orderpack::inssor
- interface mo_orderpack::omedian
- interface mo_orderpack::mrgref
- interface mo_orderpack::mrgrnk
- interface mo_orderpack::mulcnt
- interface mo_orderpack::rapknr
- interface mo_orderpack::refpar
- interface mo_orderpack::refsor
- interface mo_orderpack::rinpar
- interface mo_orderpack::rnkpar
- interface mo_orderpack::uniinv
- interface mo_orderpack::nearless
- interface mo_orderpack::unipar
- interface mo_orderpack::unirnk
- interface mo_orderpack::unista
- interface mo_orderpack::valmed
- interface mo_orderpack::valnth

Modules

- module mo_orderpack

  Sort and ranking routines.

Functions/Subroutines

- integer(i4) function, dimension(size(arr)) mo_orderpack::sort_index_dp (arr)
- integer(i4) function, dimension(size(arr)) mo_orderpack::sort_index_sp (arr)
- integer(i4) function, dimension(size(arr)) mo_orderpack::sort_index_i4 (arr)
- subroutine, private mo_orderpack::d_ctrper (XDONT, PCLS)
- subroutine, private mo_orderpack::r_ctrper (XDONT, PCLS)
- subroutine, private mo_orderpack::i_ctrper (XDONT, PCLS)
- real(kind=dp) function, private mo_orderpack::d_fndnth (XDONT, NORD)
- real(kind=sp) function, private mo_orderpack::r_fndnth (XDONT, NORD)
- integer(kind=i4) function, private mo_orderpack::i_fndnth (XDONT, NORD)
- subroutine, private mo_orderpack::d_indmed (XDONT, INDM)
- recursive subroutine, private mo_orderpack::d_med (XDATT, IDATT, ires_med)
- subroutine, private mo_orderpack::r_indmed (XDONT, INDM)
- recursive subroutine, private mo_orderpack::r_med (XDATT, IDATT, ires_med)
- subroutine, private mo_orderpack::i_indmed (XDONT, INDM)
- recursive subroutine, private mo_orderpack::i_med (XDATT, IDATT, ires_med)
- integer(kind=i4) function, private mo_orderpack::d_indnth (XDONT, NORD)

- integer(kind=i4) function, private mo_orderpack::r_indnth (XDONT, NORD)
- integer(kind=i4) function, private mo_orderpack::i_indnth (XDONT, NORD)
- subroutine, private mo_orderpack::d_inspar (XDONT, NORD)
- subroutine, private mo_orderpack::r_inspar (XDONT, NORD)
- subroutine, private mo_orderpack::i_inspar (XDONT, NORD)
- subroutine, private mo_orderpack::d_inssor (XDONT)
- subroutine, private mo_orderpack::r_inssor (XDONT)
- subroutine, private mo_orderpack::i_inssor (XDONT)
- real(kind=dp) function, private mo_orderpack::d_median (XDONT)
- real(kind=sp) function, private mo_orderpack::r_median (XDONT)
- integer(kind=i4) function, private mo_orderpack::i_median (XDONT)
- subroutine, private mo_orderpack::d_mrgref (XVALT, IRNGT)
- subroutine, private mo_orderpack::r_mrgref (XVALT, IRNGT)
- subroutine, private mo_orderpack::i_mrgref (XVALT, IRNGT)
- subroutine, private mo_orderpack::d_mrgrnk (XDONT, IRNGT)
- subroutine, private mo_orderpack::r_mrgrnk (XDONT, IRNGT)
- subroutine, private mo_orderpack::i_mrgrnk (XDONT, IRNGT)
- subroutine, private mo_orderpack::d_mulcnt (XDONT, IMULT)
- subroutine, private mo_orderpack::r_mulcnt (XDONT, IMULT)
- subroutine, private mo_orderpack::i_mulcnt (XDONT, IMULT)
- subroutine, private mo_orderpack::d_rapknr (XDONT, IRNGT, NORD)
- subroutine, private mo_orderpack::r_rapknr (XDONT, IRNGT, NORD)
- subroutine, private mo_orderpack::i_rapknr (XDONT, IRNGT, NORD)
- subroutine, private mo_orderpack::d_refpar (XDONT, IRNGT, NORD)
- subroutine, private mo_orderpack::r_refpar (XDONT, IRNGT, NORD)
- subroutine, private mo_orderpack::i_refpar (XDONT, IRNGT, NORD)
- subroutine, private mo_orderpack::d_refsor (XDONT)
- recursive subroutine, private mo_orderpack::d_subsor (XDONT, IDEB1, IFIN1)
- subroutine, private mo_orderpack::r_refsor (XDONT)
- recursive subroutine, private mo_orderpack::r_subsor (XDONT, IDEB1, IFIN1)
- subroutine, private mo_orderpack::i_refsor (XDONT)
- recursive subroutine, private mo_orderpack::i_subsor (XDONT, IDEB1, IFIN1)
- subroutine, private mo_orderpack::d_rinpar (XDONT, IRNGT, NORD)
- subroutine, private mo_orderpack::r_rinpar (XDONT, IRNGT, NORD)
- subroutine, private mo_orderpack::i_rinpar (XDONT, IRNGT, NORD)
- subroutine, private mo_orderpack::d_rnkpar (XDONT, IRNGT, NORD)
- subroutine, private mo_orderpack::r_rnkpar (XDONT, IRNGT, NORD)
- subroutine, private mo_orderpack::i_rnkpar (XDONT, IRNGT, NORD)
- subroutine, private mo_orderpack::d_uniinv (XDONT, IGOEST)
- subroutine, private mo_orderpack::r_uniinv (XDONT, IGOEST)
- subroutine, private mo_orderpack::i_uniinv (XDONT, IGOEST)
- real(kind=dp) function, private mo_orderpack::d_nearless (XVAL)
- real(kind=sp) function, private mo_orderpack::r_nearless (XVAL)
- integer(kind=i4) function, private mo_orderpack::i_nearless (XVAL)
- subroutine, private mo_orderpack::d_unipar (XDONT, IRNGT, NORD)
- subroutine, private mo_orderpack::r_unipar (XDONT, IRNGT, NORD)
- subroutine, private mo_orderpack::i_unipar (XDONT, IRNGT, NORD)
- subroutine, private mo_orderpack::d_unirnk (XVALT, IRNGT, NUNI)
- subroutine, private mo_orderpack::r_unirnk (XVALT, IRNGT, NUNI)
- subroutine, private mo_orderpack::i_unirnk (XVALT, IRNGT, NUNI)
- subroutine, private mo_orderpack::d_unista (XDONT, NUNI)
- subroutine, private mo_orderpack::r_unista (XDONT, NUNI)
- subroutine, private mo_orderpack::i_unista (XDONT, NUNI)
- recursive real(kind=dp) function, private mo_orderpack::d_valmed (XDONT)
- recursive real(kind=sp) function, private mo_orderpack::r_valmed (XDONT)

- recursive integer(kind=i4) function, private mo_orderpack::i_valmed (XDONT)
- real(kind=dp) function, private mo_orderpack::d_valnth (XDONT, NORD)
- real(kind=sp) function, private mo_orderpack::r_valnth (XDONT, NORD)
- integer(kind=i4) function, private mo_orderpack::i_valnth (XDONT, NORD)

Variables

- integer(kind=i4), dimension(:), allocatable, save mo_orderpack::idont

18.84 mo_percentile.f90 File Reference

Data Types

- interface mo_percentile::median
- interface mo_percentile::n_element
- interface mo_percentile::percentile
- interface mo_percentile::qmedian

Modules

- module mo_percentile

Functions/Subroutines

- real(dp) function mo_percentile::median_dp (arrin, mask)
- real(sp) function mo_percentile::median_sp (arrin, mask)
- real(dp) function mo_percentile::n_element_dp (idat, n, mask, before, after, previous, next)
- real(sp) function mo_percentile::n_element_sp (idat, n, mask, before, after, previous, next)
- real(dp) function mo_percentile::percentile_0d_dp (arrin, k, mask, mode_in)
- real(sp) function mo_percentile::percentile_0d_sp (arrin, k, mask, mode_in)
- real(dp) function, dimension(size(k)) mo_percentile::percentile_1d_dp (arrin, k, mask, mode_in)
- real(sp) function, dimension(size(k)) mo_percentile::percentile_1d_sp (arrin, k, mask, mode_in)
- real(dp) function mo_percentile::qmedian_dp (dat)
- real(sp) function mo_percentile::qmedian_sp (dat)

18.85 mo_pet.f90 File Reference

Modules

- module mo_pet

  Module for calculating reference/potential evapotranspiration [mm s-1].

Functions/Subroutines

- elemental pure real(dp) function, public mo_pet::pet_hargreaves (HarSamCoeff, HarSamConst, tavg, tmax, tmin, latitude, doy)

  Reference Evapotranspiration after Hargreaves.
- elemental pure real(dp) function, public mo_pet::pet_priestly (PrieTayParam, Rn, tavg)

  Reference Evapotranspiration after Priestly-Taylor.

- elemental pure real(dp) function, public mo_pet::pet_penman (net_rad, tavg, act_vap_pressure, aerodyn_↩ resistance, bulksurface_resistance, a_s, a_sh)

  Reference Evapotranspiration after Penman-Monteith.
- elemental pure real(dp) function, private mo_pet::extraterr_rad_approx (doy, latitude)

  Approximation of extraterrestrial radiation.
- elemental pure real(dp) function, private mo_pet::slope_satpressure (tavg)

  slope of saturation vapour pressure curve
- elemental pure real(dp) function, private mo_pet::sat_vap_pressure (tavg)

  calculation of the saturation vapour pressure

18.86 mo_prepare_gridded_lai.f90 File Reference

Modules

- module mo_prepare_gridded_lai

  Prepare daily LAI fields (e.g., MODIS data) for mHM.

Functions/Subroutines

- subroutine, public mo_prepare_gridded_lai::prepare_gridded_daily_lai_data (iBasin, nrows, ncols, mask, L↩ AIPer_iBasin)

  Prepare gridded daily LAI data.
- subroutine, public mo_prepare_gridded_lai::prepare_gridded_mean_monthly_lai_data (iBasin, nrows, ncols, mask)

  prepare_gridded_mean_monthly_LAI_data

18.87 mo_read_forcing_nc.f90 File Reference

Modules

- module mo_read_forcing_nc

  Reads forcing input data.

Functions/Subroutines

- subroutine, public mo_read_forcing_nc::read_forcing_nc (folder, nRows, nCols, varName, mask, data, target_period, lower, upper, nctimestep, fileName, nocheck, maskout)

  Reads forcing input in NetCDF file format.
- subroutine, public mo_read_forcing_nc::read_const_forcing_nc (folder, nRows, nCols, varName, mask, data)
- subroutine, public mo_read_forcing_nc::read_weights_nc (folder, nRows, nCols, varName, data, mask, lower, upper, nocheck, maskout, fileName)

  Reads weights for meteo forcings input in NetCDF file format.
- subroutine mo_read_forcing_nc::get_time_vector_and_select (var, fname, inctimestep, time_start, time_cnt, target_period)

  TODO: add description.

18.88 mo_read_latlon.f90 File Reference

Modules

- module mo_read_latlon

  reading latitude and longitude coordinates for each basin

Functions/Subroutines

- subroutine, public mo_read_latlon::read_latlon (ii, lon_var_name, lat_var_name, level_name, level)

  reads latitude and longitude coordinates

18.89 mo_read_lut.f90 File Reference

Modules

- module mo_read_lut

  Routines reading lookup tables (lut).

Functions/Subroutines

- subroutine, public mo_read_lut::read_geoformation_lut (filename, fileunit, nGeo, geo_unit, geo_karstic)

  Reads LUT containing geological formation information.
- subroutine, public mo_read_lut::read_lai_lut (filename, fileunit, nLAI, LAIIDlist, LAI)

  Reads LUT containing LAI information.

18.90 mo_read_optional_data.f90 File Reference

Modules

- module mo_read_optional_data

  Read optional data for mHM calibration.

Functions/Subroutines

- subroutine, public mo_read_optional_data::read_soil_moisture (iBasin)

  Read soil moisture data from NetCDF file for calibration.
- subroutine, public mo_read_optional_data::read_basin_avg_tws

  Read basin average TWS timeseries from file, the same way runoff is read.
- subroutine, public mo_read_optional_data::read_neutrons (iBasin)

  Read neutrons data from NetCDF file for calibration.
- subroutine, public mo_read_optional_data::read_evapotranspiration (iBasin)

  Read evapotranspiration data from NetCDF file for calibration.

18.91 mo_read_spatial_data.f90 File Reference

Data Types

- interface mo_read_spatial_data::read_spatial_data_ascii

  Reads spatial data files of ASCII format.

Modules

- module mo_read_spatial_data

  Reads spatial input data.

Functions/Subroutines

- subroutine mo_read_spatial_data::read_spatial_data_ascii_dp (filename, fileunit, header_ncols, header_nrows, header_xllcorner, header_yllcorner, header_cellsize, data, mask)

  TODO: add description.
- subroutine mo_read_spatial_data::read_spatial_data_ascii_i4 (filename, fileunit, header_ncols, header_nrows, header_xllcorner, header_yllcorner, header_cellsize, data, mask)

  TODO: add description.
- subroutine, public mo_read_spatial_data::read_header_ascii (filename, fileunit, header_ncols, header_nrows, header_xllcorner, header_yllcorner, header_cellsize, header_nodata)

  Reads header lines of ASCII files.

18.92 mo_read_timeseries.f90 File Reference

Modules

- module mo_read_timeseries

  Routines to read files containing timeseries data.

Functions/Subroutines

- subroutine, public mo_read_timeseries::read_timeseries (filename, fileunit, periodStart, periodEnd, optimize, opti_function, data, mask, nMeasPerDay)

  Reads time series in ASCII format.

18.93 mo_read_wrapper.f90 File Reference

Modules

- module mo_read_wrapper

  Wrapper for all reading routines.

Functions/Subroutines

- subroutine, public mo_read_wrapper::read_data (LAIPer)

  Reads data.
- subroutine mo_read_wrapper::check_consistency_lut_map (data, lookuptable, filename, unique_values)

  Checks if classes in input maps appear in look up tables.

18.94 mo_restart.f90 File Reference

Data Types

- interface mo_restart::unpack_field_and_write

  TODO: add description.

Modules

- module mo_restart

  reading and writing states, fluxes and configuration for restart of mHM.

Functions/Subroutines

- subroutine, public mo_restart::write_restart_files (OutPath)

  write restart files for each basin
- subroutine, public mo_restart::read_restart_states (iBasin, InPath)

  reads fluxes and state variables from file
- subroutine mo_restart::unpack_field_and_write_1d_i4 (nc, var_name, var_dims, fill_value, data, mask, var↩_long_name)
- subroutine mo_restart::unpack_field_and_write_1d_dp (nc, var_name, var_dims, fill_value, data, mask, var↩_long_name)
- subroutine mo_restart::unpack_field_and_write_2d_dp (nc, var_name, var_dims, fill_value, data, mask, var↩_long_name)
- subroutine mo_restart::unpack_field_and_write_3d_dp (nc, var_name, var_dims, fill_value, data, mask, var↩_long_name)

18.95 mo_runoff.f90 File Reference

Modules

- module mo_runoff

  Runoff generation for the unsaturated zone, saturated zone (or groundwater zone), and runoff accumulation.

Functions/Subroutines

- subroutine, public mo_runoff::runoff_unsat_zone (k1, kp, k0, alpha, karst_loss, pefec_soil, unsat_thresh, sat_storage, unsat_storage, slow_interflow, fast_interflow, perc)

  Runoff generation for the saturated zone.
- subroutine, public mo_runoff::runoff_sat_zone (k2, sat_storage, baseflow)

  Runoff generation for the saturated zone.
- subroutine, public mo_runoff::l1_total_runoff (fSealed_area_fraction, fast_interflow, slow_interflow, baseflow, direct_runoff, total_runoff)

  total runoff accumulation at level 1

18.96 mo_sce.f90 File Reference

Modules

- module mo_sce

  Shuffled Complex Evolution optimization algorithm.

Functions/Subroutines

- real(dp) function, dimension(size(pini, 1)), public mo_sce::sce (eval, functn, pini, prange, mymaxn, mymaxit, mykstop, mypcento, mypeps, myseed, myngs, mynpg, mynps, mynspl, mymings, myiniflg, myprint, mymask, myalpha, mybeta, tmp_file, popul_file, popul_file_append, parallel, restart, restart_file, bestf, neval, history)

  Shuffled Complex Evolution (SCE) algorithm for global optimization.
- subroutine write_best_intermediate (to_file)
- subroutine write_best_final ()
- subroutine write_population (to_file)
- subroutine write_termination_case (case)
- subroutine set_optional ()
- subroutine mo_sce::parstt (x, bound, peps, mask, xnstd, gnrng, ipcnvg)
- subroutine mo_sce::comp (ngs2, npg, a, af, b, bf)
- subroutine mo_sce::sort_matrix (rb, ra)
- subroutine mo_sce::chkcst (x, bl, bu, mask, ibound)
- subroutine mo_sce::getpnt (idist, bl, bu, std, xi, mask, save_state, x)
- subroutine mo_sce::cce (s, sf, bl, bu, maskpara, xnstd, icall, maxn, maxit, save_state_gauss, functn, eval, alpha, beta, history, idot)

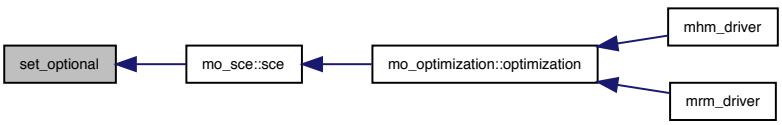
18.96.1 Function/Subroutine Documentation

18.96.1.1 set_optional()

```
subroutine sce::set_optional ( )  [private]
```

Referenced by mo_sce::sce().

Here is the caller graph for this function:

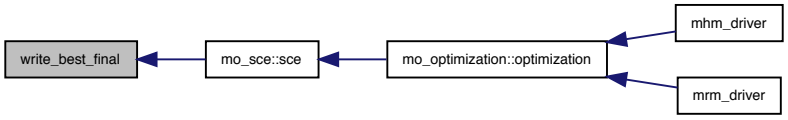


18.96.1.2 write_best_final()

```
subroutine sce::write_best_final ( )  [private]
```

Referenced by mo_sce::sce().

Here is the caller graph for this function:

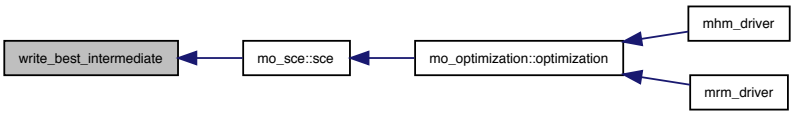


18.96.1.3 write_best_intermediate()

```
subroutine sce::write_best_intermediate (
            logical, intent(in) to_file )
```

Referenced by mo_sce::sce().

Here is the caller graph for this function:

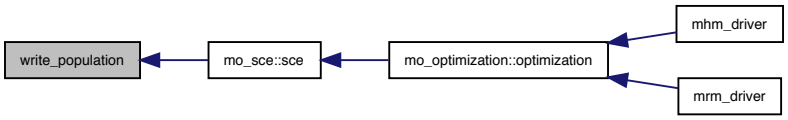


18.96.1.4 write_population()

```
subroutine sce::write_population (
            logical, intent(in) to_file )  [private]
```

Referenced by mo_sce::sce().

Here is the caller graph for this function:

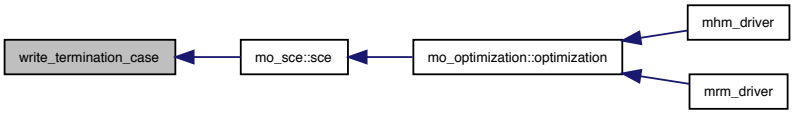


18.96.1.5 write_termination_case()

```
subroutine sce::write_termination_case (
            integer(i4), intent(in) case )  [private]
```

Referenced by mo_sce::sce().

Here is the caller graph for this function:



18.97 mo_set_netcdf_outputs.f90 File Reference

Modules

- module mo_set_netcdf_outputs

  Defines the structure of the netCDF to write the output in.

Functions/Subroutines

- subroutine, public mo_set_netcdf_outputs::set_netcdf (NoNetcdfVars, nrows, ncols)

  Runs mhm with a specific parameter set and returns required variables, e.g. runoff.

18.98 mo_snow_accum_melt.f90 File Reference

Modules

- module mo_snow_accum_melt

  Snow melting and accumulation.

Functions/Subroutines

- subroutine, public mo_snow_accum_melt::snow_accum_melt (deg_day_incr, deg_day_max, deg_day_↩ noprec, prec, temperature, temperature_thresh, thrfall, snow_pack, deg_day, melt, prec_effect, rain, snow)

  Snow melting and accumulation.

18.99 mo_soil_database.f90 File Reference

Modules

- module mo_soil_database

  Generating soil database from input file.

Functions/Subroutines

- subroutine, public mo_soil_database::read_soil_lut (filename)

  Reads the soil LUT file.
- subroutine, public mo_soil_database::generate_soil_database

  Generates soil database.

18.100 mo_soil_moisture.f90 File Reference

Modules

- module mo_soil_moisture

  Soil moisture of the different layers.

Functions/Subroutines

- subroutine, public mo_soil_moisture::soil_moisture (processCase, frac_sealed, water_thresh_sealed, pet, evap_coeff, soil_moist_sat, frac_roots, soil_moist_FC, wilting_point, soil_moist_exponen, jarvis_thresh_c1, aet_canopy, prec_effec, runoff_sealed, storage_sealed, infiltration, soil_moist, aet, aet_sealed)

  Soil moisture in different soil horizons.
- elemental pure real(dp) function, private mo_soil_moisture::feddes_et_reduction (soil_moist, soil_moist_FC, wilting_point, frac_roots)

  stress factor for reducing evapotranspiration based on actual soil moisture
- elemental pure real(dp) function, private mo_soil_moisture::jarvis_et_reduction (soil_moist, soil_moist_sat, wilting_point, frac_roots, jarvis_thresh_c1)

  stress factor for reducing evapotranspiration based on actual soil moisture

18.101 mo_spatial_agg_disagg_forcing.f90 File Reference

Data Types

- interface mo_spatial_agg_disagg_forcing::spatial_aggregation

  Spatial aggregation of meterological variables.
- interface mo_spatial_agg_disagg_forcing::spatial_disaggregation

  Spatial disaggregation of meterological variables.

Modules

- module mo_spatial_agg_disagg_forcing

  Spatial aggegation or disaggregation of meteorological input data.

Functions/Subroutines

- subroutine mo_spatial_agg_disagg_forcing::spatial_aggregation_3d (data2, cellsize2, cellsize1, mask1, mask2, data1)
- subroutine mo_spatial_agg_disagg_forcing::spatial_aggregation_4d (data2, cellsize2, cellsize1, mask1, mask2, data1)
- subroutine mo_spatial_agg_disagg_forcing::spatial_disaggregation_3d (data2, cellsize2, cellsize1, mask1, mask2, data1)
- subroutine mo_spatial_agg_disagg_forcing::spatial_disaggregation_4d (data2, cellsize2, cellsize1, mask1, mask2, data1)

18.102 mo_spatialsimilarity.f90 File Reference

Data Types

- interface mo_spatialsimilarity::nndv

  Calculates the number of neighboring dominating values, a measure for spatial dissimilarity.
- interface mo_spatialsimilarity::pd

  Calculates pattern dissimilarity (PD) measure.

Modules

- module mo_spatialsimilarity

  Routines for bias insensitive comparison of spatial patterns.

Functions/Subroutines

- real(sp) function mo_spatialsimilarity::nndv_sp (mat1, mat2, mask, valid)
- real(dp) function mo_spatialsimilarity::nndv_dp (mat1, mat2, mask, valid)
- real(sp) function mo_spatialsimilarity::pd_sp (mat1, mat2, mask, valid)
- real(dp) function mo_spatialsimilarity::pd_dp (mat1, mat2, mask, valid)

18.103 mo_standard_score.f90 File Reference

Data Types

- interface mo_standard_score::standard_score

  Calculates the standard score / normalization (anomaly) / z-score.
- interface mo_standard_score::classified_standard_score

  Calculates the classified standard score (e.g. classes are months).

Modules

- module mo_standard_score

  Routines for calculating the normalization (anomaly)/standard score/z score and the deseasonalized (standard score on monthly basis) values of a time series.

Functions/Subroutines

- real(sp) function, dimension(size(data, dim=1)) mo_standard_score::standard_score_sp (data, mask)
- real(dp) function, dimension(size(data, dim=1)) mo_standard_score::standard_score_dp (data, mask)
- real(sp) function, dimension(size(data, dim=1)) mo_standard_score::classified_standard_score_sp (data, classes, mask)
- real(dp) function, dimension(size(data, dim=1)) mo_standard_score::classified_standard_score_dp (data, classes, mask)

18.104 mo_startup.f90 File Reference

Modules

- module mo_startup

  Startup procedures for mHM.

Functions/Subroutines

- subroutine, public mo_startup::mhm_initialize

  Initialize main mHM variables.
- subroutine mo_startup::constants_init

  Initialize mHM constants.
- subroutine mo_startup::l2_variable_init (iBasin, level0_iBasin, level2_iBasin)

  Initalize Level-2 meteorological forcings data.

18.105 mo_string_utils.f90 File Reference

Data Types

- interface mo_string_utils::num2str

  Convert to string.
- interface mo_string_utils::numarray2str

  Convert to string.

Modules

- module mo_string_utils

  String utilities.

Functions/Subroutines

- character(len(whitespaces)) function, public mo_string_utils::compress (whiteSpaces, n)
- subroutine, public mo_string_utils::divide_string (string, delim, strArr)

  Divide string in substrings.
- logical function, public mo_string_utils::equalstrings (string1, string2)
- logical function, public mo_string_utils::nonull (str)

  Checks if string was already used.
- character(len=256) function, dimension(:), allocatable, public mo_string_utils::splitstring (string, delim)
- logical function, public mo_string_utils::startswith (string, start)
- character(len=len_trim(upper)) function, public mo_string_utils::tolower (upper)

  Convert to lower case.
- character(len=len_trim(lower)) function, public mo_string_utils::toupper (lower)
- pure character(len=10) function mo_string_utils::i42str (nn, form)
- pure character(len=20) function mo_string_utils::i82str (nn, form)
- pure character(len=32) function mo_string_utils::sp2str (rr, form)
- pure character(len=32) function mo_string_utils::dp2str (rr, form)
- pure character(len=10) function mo_string_utils::log2str (ll, form)
- character(len=size(arr)) function mo_string_utils::i4array2str (arr)
- integer(i4) function, dimension(:), allocatable, public mo_string_utils::str2num (string)

Variables

- character(len=∗), parameter, public mo_string_utils::separator = repeat('-', 70)

18.106 mo_template.f90 File Reference

Data Types

- interface mo_template::mean

  The average.

Modules

- module mo_template

  Template for future module developments.

Functions/Subroutines

- elemental pure real(dp) function, public mo_template::circum (radius)

  Circumference of a circle.
- real(dp) function mo_template::mean_dp (dat, mask)
- real(sp) function mo_template::mean_sp (dat, mask)

Variables

- real(dp), parameter, public mo_template::pi_dp = 3.14159265358979323846264338327950288419 7_dp

  Constant Pi in double precision.
- real(sp), parameter, public mo_template::pi_sp = 3.14159265358979323846264338327950288419 7_sp

  Constant Pi in single precision.
- integer(i4), parameter mo_template::itest = 1

18.107 mo_temporal_aggregation.f90 File Reference

Data Types

- interface mo_temporal_aggregation::day2mon_average

  Day-to-month average (day2mon_average)
- interface mo_temporal_aggregation::hour2day_average

  Hour-to-day average (hour2day_average)

Modules

- module mo_temporal_aggregation

  Temporal aggregation for time series (averaging)

Functions/Subroutines

- subroutine mo_temporal_aggregation::day2mon_average_dp (daily_data, yearS, monthS, dayS, mon_avg, misval, rm_misval)
- subroutine mo_temporal_aggregation::hour2day_average_dp (hourly_data, yearS, monthS, dayS, hourS, day_avg, misval, rm_misval)

18.108 mo_temporal_disagg_forcing.f90 File Reference

Modules

- module mo_temporal_disagg_forcing

  Temporal disaggregation of daily input values.

Functions/Subroutines

- elemental pure subroutine, public mo_temporal_disagg_forcing::temporal_disagg_forcing (isday, ntimesteps↩ _day, prec_day, pet_day, temp_day, fday_prec, fday_pet, fday_temp, fnight_prec, fnight_pet, fnight_temp, temp_weights, pet_weights, pre_weights, read_meteo_weights, prec, pet, temp)

  Temporally distribute daily mean forcings onto time step.

18.109 mo_timer.f90 File Reference

Modules

- module mo_timer

  Timing routines.

Functions/Subroutines

- subroutine, public mo_timer::timer_check (timer)

  Check a timer.
- subroutine, public mo_timer::timer_clear (timer)

  Reset a timer.
- real(sp) function, public mo_timer::timer_get (timer)

  Return a timer.
- subroutine, public mo_timer::timer_print (timer)

  Print a timer.
- subroutine, public mo_timer::timer_start (timer)

  Start a timer.
- subroutine, public mo_timer::timer_stop (timer)

  Stop a timer.
- subroutine, public mo_timer::timers_init

  Initialise timer module.

Variables

- integer(i4), parameter, public mo_timer::max_timers = 99

  max number of timers allowed
- integer(i4), save, public mo_timer::cycles_max

  max value of clock allowed by system
- real(sp), save, public mo_timer::clock_rate

  clock_rate in seconds for each cycle
- integer(i4), dimension(max_timers), save, public mo_timer::cycles1

  cycle number at start for each timer
- integer(i4), dimension(max_timers), save, public mo_timer::cycles2

cycle number at stop for each timer

- real(sp), dimension(max_timers), save, public mo_timer::cputime

  accumulated cpu time in each timer

- character(len=8), dimension(max_timers), save, public mo_timer::status

  timer status string

18.110 mo_upscaling_operators.f90 File Reference

Modules

- module mo_upscaling_operators

  Module containing upscaling operators.

Functions/Subroutines

- integer(i4) function, dimension(size(l1_upper_rowid_cell, 1)), public mo_upscaling_operators::majority_statistics (nClass, L1_upper_rowId_cell, L1_lower_rowId_cell, L1_left_colonId_cell, L1_right_colonId_cell, L0_fineScale_2D_data)

  majority statistics

- real(dp) function, dimension(size(l0upbound_inlx, 1)), public mo_upscaling_operators::l0_fractionalcover_in_lx (dataIn0, classId, mask0, L0upBound_inLx, L0downBound_inLx, L0leftBound_inLx, L0rightBound_inLx, nTCells0_inLx)

  fractional coverage of a given class of L0 fields in Lx field (Lx = L1 or L11)

- real(dp) function, dimension(size(nl0_cells_in_l1_cell, 1)), public mo_upscaling_operators::upscale_arithmetic_mean (nL0_cells_in_L1_cell, L1_upper_rowId_cell, L1_lower_rowId_cell, L1_left_colonId_cell, L1_right_colonId_cell, L0_cellId, mask0, nodata_value, L0_fineScale_data)

  aritmetic mean

- real(dp) function, dimension(size(nl0_cells_in_l1_cell, 1)), public mo_upscaling_operators::upscale_harmonic_mean (nL0_cells_in_L1_cell, L1_upper_rowId_cell, L1_lower_rowId_cell, L1_left_colonId_cell, L1_right_colonId_cell, L0_cellId, mask0, nodata_value, L0_fineScale_data)

  harmonic mean

- real(dp) function, dimension(size(l1_upper_rowid_cell, 1)), public mo_upscaling_operators::upscale_geometric_mean (L1_upper_rowId_cell, L1_lower_rowId_cell, L1_left_colonId_cell, L1_right_colonId_cell, mask0, nodata_value, L0_fineScale_data)

  geometric mean

- real(dp) function, dimension(size(nl0_cells_in_l1_cell, 1)) mo_upscaling_operators::upscale_p_norm (nL0_cells_in_L1_cell, L1_upper_rowId_cell, L1_lower_rowId_cell, L1_left_colonId_cell, L1_right_colonId_cell, L0_cellId, mask0, nodata_value, p_norm, L0_fineScale_data)

  aritmetic mean

18.111 mo_utils.f90 File Reference

Data Types

- interface mo_utils::equal

  Comparison of real values.

- interface mo_utils::notequal
- interface mo_utils::greaterequal
- interface mo_utils::lesserequal
- interface mo_utils::eq
- interface mo_utils::ne

- interface mo_utils::ge
- interface mo_utils::le
- interface mo_utils::is_finite

  .true. if not IEEE Inf, IEEE NaN, nor IEEE Inf nor IEEE NaN, respectively.
- interface mo_utils::is_nan
- interface mo_utils::is_normal
- interface mo_utils::locate

  Find closest values in a monotonic series, returns the indexes.
- interface mo_utils::swap

  Swap to values or two elements in array.
- interface mo_utils::special_value

  Special IEEE values.

Modules

- module mo_utils

  General utilities for the CHS library.

Functions/Subroutines

- elemental pure logical function mo_utils::equal_dp (a, b)
- elemental pure logical function mo_utils::equal_sp (a, b)
- elemental pure logical function mo_utils::greaterequal_dp (a, b)
- elemental pure logical function mo_utils::greaterequal_sp (a, b)
- elemental pure logical function mo_utils::lesserequal_dp (a, b)
- elemental pure logical function mo_utils::lesserequal_sp (a, b)
- elemental pure logical function mo_utils::notequal_dp (a, b)
- elemental pure logical function mo_utils::notequal_sp (a, b)
- elemental pure logical function mo_utils::is_finite_dp (a)
- elemental pure logical function mo_utils::is_finite_sp (a)
- elemental pure logical function mo_utils::is_nan_dp (a)
- elemental pure logical function mo_utils::is_nan_sp (a)
- elemental pure logical function mo_utils::is_normal_dp (a)
- elemental pure logical function mo_utils::is_normal_sp (a)
- integer(i4) function mo_utils::locate_0d_dp (x, y)
- integer(i4) function mo_utils::locate_0d_sp (x, y)
- integer(i4) function, dimension(:), allocatable mo_utils::locate_1d_dp (x, y)
- integer(i4) function, dimension(:), allocatable mo_utils::locate_1d_sp (x, y)
- elemental pure subroutine mo_utils::swap_xy_dp (x, y)
- elemental pure subroutine mo_utils::swap_xy_sp (x, y)
- elemental pure subroutine mo_utils::swap_xy_i4 (x, y)
- subroutine mo_utils::swap_vec_dp (x, i1, i2)
- subroutine mo_utils::swap_vec_sp (x, i1, i2)
- subroutine mo_utils::swap_vec_i4 (x, i1, i2)
- real(dp) function mo_utils::special_value_dp (x, ieee)
- real(sp) function mo_utils::special_value_sp (x, ieee)

18.112 mo_write_ascii.f90 File Reference

Modules

- module mo_write_ascii

  Module to write ascii file output.

Functions/Subroutines

- subroutine, public mo_write_ascii::write_configfile

  This modules writes the results of the configuration into an ASCII-file.
- subroutine, public mo_write_ascii::write_optifile (best_OF, best_paramSet, param_names)

  Write briefly final optimization results.
- subroutine, public mo_write_ascii::write_optinamelist (processMatrix, parameters, maskpara, parameters_↩ name)

  Write final, optimized parameter set in a namelist format.

18.113 mo_write_fluxes_states.f90 File Reference

Data Types

- interface mo_write_fluxes_states::outputvariable
- interface mo_write_fluxes_states::outputvariable
- interface mo_write_fluxes_states::outputdataset
- interface mo_write_fluxes_states::outputdataset

Modules

- module mo_write_fluxes_states

  Creates NetCDF output for different fluxes and state variables of mHM.

Functions/Subroutines

- type(outputvariable) function mo_write_fluxes_states::newoutputvariable (nc, name, dtype, dims, ncells, mask, avg)

  Initialize OutputVariable.
- subroutine mo_write_fluxes_states::updatevariable (self, data)

  Update OutputVariable.
- subroutine mo_write_fluxes_states::writevariabletimestep (self, timestep)

  Write timestep to file.
- type(outputdataset) function, public mo_write_fluxes_states::newoutputdataset (ibasin, mask1, nCells)

  Initialize OutputDataset.
- subroutine mo_write_fluxes_states::updatedataset (self, sidx, eidx, L1_fSealed, L1_fNotSealed, L1_inter, L1_snowPack, L1_soilMoist, L1_soilMoistSat, L1_sealSTW, L1_unsatSTW, L1_satSTW, L1_neutrons, L1_↩ pet, L1_aETSoil, L1_aETCanopy, L1_aETSealed, L1_total_runoff, L1_runoffSeal, L1_fastRunoff, L1_slow↩ Runoff, L1_baseflow, L1_percol, L1_infilSoil, L1_preEffect)

  Update all variables.
- subroutine mo_write_fluxes_states::writetimestep (self, timestep)

  Write all accumulated data.
- subroutine mo_write_fluxes_states::close (self)

  Close the file.
- type(ncdataset) function mo_write_fluxes_states::createoutputfile (ibasin)

  Create and initialize output file.
- subroutine mo_write_fluxes_states::writevariableattributes (var, long_name, unit)

  Write output variable attributes.
- character(16) function mo_write_fluxes_states::fluxesunit (ibasin)

  Generate a unit string.

18.114 mo_xor4096.f90 File Reference

Data Types

- interface mo_xor4096::get_timeseed
- interface mo_xor4096::xor4096
- interface mo_xor4096::xor4096g

Modules

- module mo_xor4096

Functions/Subroutines

- subroutine mo_xor4096::get_timeseed_i4_0d (seed)
- subroutine mo_xor4096::get_timeseed_i4_1d (seed)
- subroutine mo_xor4096::get_timeseed_i8_0d (seed)
- subroutine mo_xor4096::get_timeseed_i8_1d (seed)
- subroutine mo_xor4096::xor4096s_0d (seed, SingleIntegerRN, save_state)
- subroutine mo_xor4096::xor4096s_1d (seed, SingleIntegerRN, save_state)
- subroutine mo_xor4096::xor4096f_0d (seed, SingleRealRN, save_state)
- subroutine mo_xor4096::xor4096f_1d (seed, SingleRealRN, save_state)
- subroutine mo_xor4096::xor4096l_0d (seed, DoubleIntegerRN, save_state)
- subroutine mo_xor4096::xor4096l_1d (seed, DoubleIntegerRN, save_state)
- subroutine mo_xor4096::xor4096d_0d (seed, DoubleRealRN, save_state)
- subroutine mo_xor4096::xor4096d_1d (seed, DoubleRealRN, save_state)
- subroutine mo_xor4096::xor4096gf_0d (seed, SingleRealRN, save_state)
- subroutine mo_xor4096::xor4096gf_1d (seed, SingleRealRN, save_state)
- subroutine mo_xor4096::xor4096gd_0d (seed, DoubleRealRN, save_state)
- subroutine mo_xor4096::xor4096gd_1d (seed, DoubleRealRN, save_state)

Variables

- integer(i4), parameter, public mo_xor4096::n_save_state = 132_i4

  Dimension of vector saving the state of a stream.

18.115 mpr_driver.f90 File Reference

Modules

- module dummy_mpr

Functions/Subroutines

- program mpr_driver

  Distributed precipitation-runoff model mHM.

18.115.1 Function/Subroutine Documentation

18.115.1.1 mpr_driver()

```
program mpr_driver ( )
```

Distributed precipitation-runoff model mHM.

This is the main driver of mHM, which calls one instance of mHM for a multiple basins and a given period.

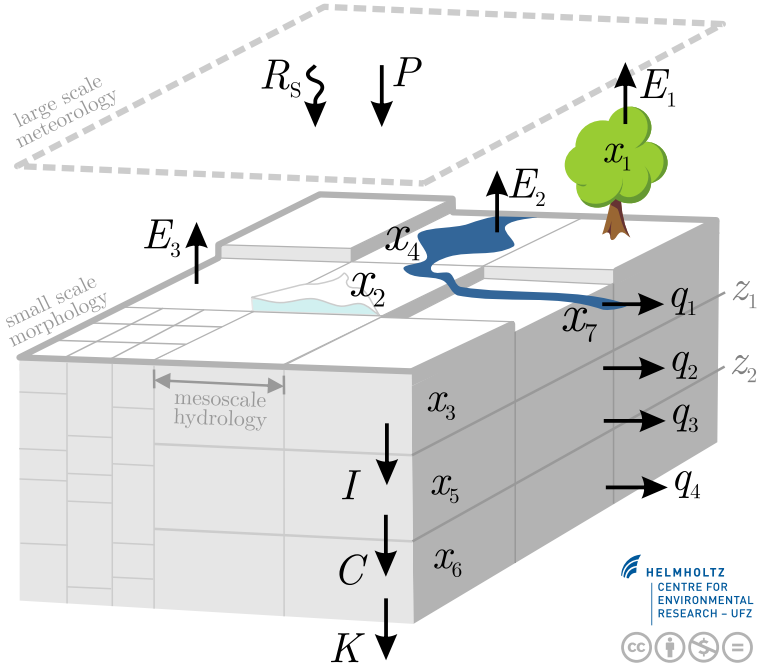


Figure 18.2: Typical mHM cell

Luis Samaniego & Rohini Kumar (UFZ)

Date

Dec 2015

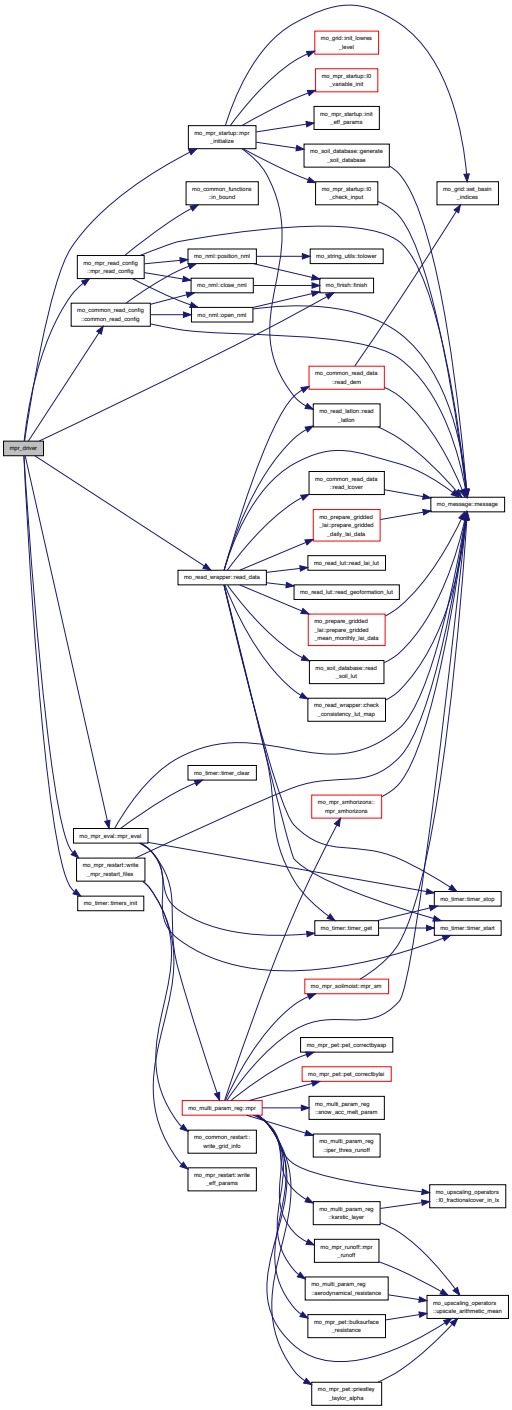
Version

0.1

Copyright

References mo_common_read_config::common_read_config(), mo_common_variables::dirrestartout, mo_kind↩ ::dp, mo_mpr_file::file_namelist_mpr_param, mo_finish::finish(), mo_mpr_eval::mpr_eval(), mo_mpr_startup::mpr↩ _initialize(), mo_mpr_read_config::mpr_read_config(), mo_read_wrapper::read_data(), mo_timer::timers_init(), mo_mpr_file::unamelist_mpr_param, mo_mpr_restart::write_mpr_restart_files(), and mo_common_variables↩ ::write_restart.

Here is the call graph for this function:



18.116 mrm_driver.f90 File Reference

Modules

- module dummy_mrm

Functions/Subroutines

- program mrm_driver

  TODO: add description.

18.116.1 Function/Subroutine Documentation

18.116.1.1 mrm_driver()

```
program mrm_driver ( )
```

TODO: add description.

TODO: add description

Authors

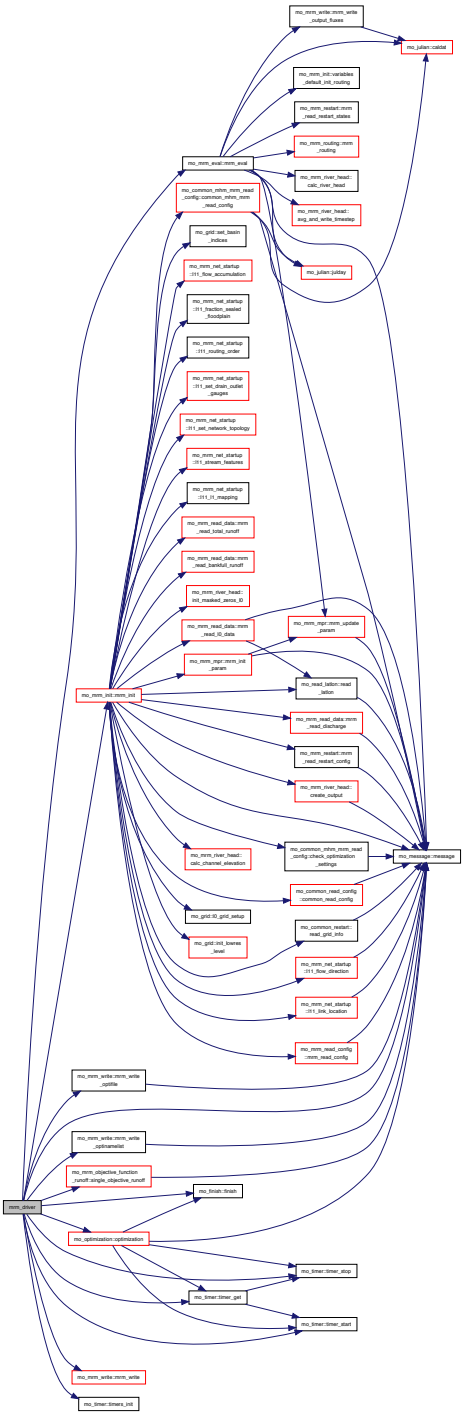
Stephan Thober

Date

Jun 2018

References mo_common_variables::dirconfigout, mo_kind::dp, mo_mrm_file::file_namelist_mrm, mo_mrm_↩file::file_namelist_param_mrm, mo_finish::finish(), mo_common_variables::global_parameters, mo_common_↩variables::global_parameters_name, mo_kind::i4, mo_message::message(), mo_common_mhm_mrm_variables↩::mrm_coupling_mode, mo_mrm_eval::mrm_eval(), mo_mrm_init::mrm_init(), mo_mrm_write::mrm_write(), mo_↩mrm_write::mrm_write_optifile(), mo_mrm_write::mrm_write_optinamelist(), mo_optimization::optimization(), mo↩_common_mhm_mrm_variables::optimize, mo_mrm_objective_function_runoff::single_objective_runoff(), mo↩_timer::timer_get(), mo_timer::timer_start(), mo_timer::timer_stop(), mo_timer::timers_init(), mo_mrm_file↩::unamelist_mrm, and mo_mrm_file::unamelist_param_mrm.

Here is the call graph for this function:



18.117 RELEASES.md File Reference

Bibliography

[1] E. Aarts and J. Korst. *Simulated annealing and Boltzmann machines: a stochastic approach to combinatorial optimization and neural computing*. Wiley, Chichester, 1990. 6

[2] S. Bergström. Development and application of a conceptual runoff model for scandinavian catchments. Technical Report 7, SMHI Reports RHO, Norrköping, 1976. 1

[3] K. Beven. Prophesy, reality and uncertainty in distributed hydrological modelling. *Adv. Water Resour.*, 16:41–51, 1993. 4

[4] G. Blöschl. Scaling in hydrology. *Hydrol. Process.*, 15(4):709–711, 2001. 2

[5] G. Blöschl, C. Reszler, and J. Komma. A spatially distributed flash flood forecasting model. *Environ. Model. Soft.*, 23(4):464–478, 2008. 1

[6] G. Blöschl and M. Sivapalan. Scale issues in hydrological modelling: A review. *Hydrol. Process.*, 9(3-4):251–290, 1995. 2

[7] V. T. Chow, D. R Maidment, and L. W. Mays. *Applied Hydrology*. McGraw-Hill, 1988. 421, 423

[8] R Courant, K Friedrichs, and H Lewy. Über die partiellen Differenzengleichungen der mathematischen Physik. *Mathematische Annalen*, 100(1):32–74, 1928. 50

[9] L. Duckstein. Multiobjective optimization in structural design: The model choice problem. In E. Atrek, R. H. Gallagher, K. M. Ragsdell, and O. C. Zienkiewicz, editors, *New Directions in Optimum Structural Design*, pages 459–481. Eds. John Wiley and Sons Inc., New York, 1984. 6

[10] G. Hartmann and A. Bárdossy. Investigation of the transferability of hydrological models and a method to improve model calibration. *Adv. Geosciences*, 5:83–87, 2005. 6

[11] Y. Hundecha and A. Bárdossy. Modeling effect of land use changes on runoff generation of a river basin through parameter regionalization of a watershed model. *J. Hydrol.*, 292:281–295, 2004. 1

[12] R. Kumar, L. Samaniego, and S. Attinger. Implications of distributed hydrologic model parametrization on the simulation of water fluxes at multiple scales and locations. In press. *Water Resour. Res.*, 2012. 51

[13] X. Liang, D. P. Lettenmaier, E. F. Wood, and S. J. Burges. A simple hydrologically based model of land-surface water and energy fluxes for general-circulation models. *J. Geophys. Res.-Atmos.*, 99(D7):14415–14428, 1994. 1

[14] P. Pokhrel and H. V. Gupta. On the use of spatial-regularization strategies to improve calibration of distributed watershed models. *Water Resour. Res.*, 2009. In press. 4

[15] L. Samaniego and A. Bárdossy. Robust parametric models of runoff characteristics at the mesoscale. *Journal of Hydrology*, 303(1-4):136–151, 2005. doi: 10.1016/j.jhydrol.2004.08.022. 365

[16] L. Samaniego, R. Kumar, and S. Attinger. Multiscale parameter regionalization of a grid-based hydrologic model at the mesoscale. *Water Resour. Res.*, 46, 2010. 2, 5, 51

[17] B. A. Tolson and C. A. Shoemaker. Dynamically dimensioned search algorithm for computationally efficient watershed model calibration. *Water Resources Research*, 43(1):W01413, 2007. 6

Index

1-main.dox, 959
2-get_started.dox, 959
3-data_preparation.dox, 959
4-visualise_out.dox, 959
5-calibration.dox, 959
6-style_guide.dox, 959
7-test_basin.dox, 959
8-protocols_for_setup_new_mHM_basin.dox, 959
9-mRM.dox, 959

absdev_dp
    mo_moment, 266
    mo_moment::absdev, 753
absdev_sp
    mo_moment, 266
    mo_moment::absdev, 753
add_inflow
    mo_mrm_routing, 420
aerodynamical_resistance
    mo_multi_param_reg, 455
all
    mo_mrm_write_fluxes_states::outputdataset, 877
    mo_write_fluxes_states::outputdataset, 881
alma_convention
    mo_common_variables, 123
anneal_dp
    mo_anneal, 83
    mo_anneal::anneal, 755
append_char_3d
    mo_append, 88
    mo_append::append, 758
append_char_m_m
    mo_append, 88
    mo_append::append, 758
append_char_v_s
    mo_append, 89
    mo_append::append, 758
append_char_v_v
    mo_append, 89
    mo_append::append, 758
append_dp_3d
    mo_append, 89
    mo_append::append, 758
append_dp_m_m
    mo_append, 89
    mo_append::append, 759
append_dp_v_s
    mo_append, 89
    mo_append::append, 759
append_dp_v_v
    mo_append, 89
    mo_append::append, 759
append_i4_3d
    mo_append, 90
append_i4_m_m
    mo_append, 90
    mo_append::append, 759
append_i4_v_s
    mo_append, 90
    mo_append::append, 759
append_i4_v_v
    mo_append, 90
    mo_append::append, 759
append_i8_3d
    mo_append, 90
    mo_append::append, 760
append_i8_m_m
    mo_append, 90
    mo_append::append, 760
append_i8_v_s
    mo_append, 90
    mo_append::append, 760
append_i8_v_v
    mo_append, 91
    mo_append::append, 760
append_lgt_3d
    mo_append, 91
    mo_append::append, 760
append_lgt_m_m
    mo_append, 91
    mo_append::append, 760
append_lgt_v_s
    mo_append, 91
    mo_append::append, 760
append_lgt_v_v
    mo_append, 91
    mo_append::append, 761
append_sp_3d
    mo_append, 91
    mo_append::append, 761
append_sp_m_m
    mo_append, 92
    mo_append::append, 761
append_sp_v_s
    mo_append, 92
    mo_append::append, 761
append_sp_v_v
    mo_append, 92
    mo_append::append, 761

approx_mon_int
mo_neutrons, 580
approx_mon_int_eps
mo_neutrons, 581
approx_mon_int_steps
mo_neutrons, 582
arth_dp
mo_corr, 140
mo_corr::arth, 762
arth_i4
mo_corr, 140
mo_corr::arth, 762
arth_sp
mo_corr, 141
mo_corr::arth, 762
att
mo_ncwrite::variable, 949
autocoeffk_1d_dp
mo_corr, 141
mo_corr::autocoeffk, 764
autocoeffk_1d_sp
mo_corr, 141
mo_corr::autocoeffk, 764
autocoeffk_dp
mo_corr, 141
mo_corr::autocoeffk, 764
autocoeffk_sp
mo_corr, 141
mo_corr::autocoeffk, 764
autocorr_1d_dp
mo_corr, 141
mo_corr::autocorr, 765
autocorr_1d_sp
mo_corr, 142
mo_corr::autocorr, 765
autocorr_dp
mo_corr, 142
mo_corr::autocorr, 765
autocorr_sp
mo_corr, 142
mo_corr::autocorr, 765
autocorrelation
mo_mrm_signatures, 428
average
mo_mrm_write_fluxes_states::outputvariable, 885
mo_write_fluxes_states::outputvariable, 890
average_counter
mo_mrm_write, 445
average_dp
mo_moment, 266
mo_moment::average, 766
average_sp
mo_moment, 266
mo_moment::average, 766
avg
mo_mrm_write_fluxes_states::outputvariable, 885
mo_write_fluxes_states::outputvariable, 890
avg_and_write_timestep
mo_mrm_river_head, 414

baseflow_param
mo_multi_param_reg, 457
basin
mo_mrm_write_fluxes_states::outputdataset, 877
mo_write_fluxes_states::outputdataset, 881
basin_avg_tws_obs
mo_global_variables, 180
basin_avg_tws_sim
mo_global_variables, 181
basin_mrm
mo_mrm_global_variables, 334
basinid
mo_global_variables::twsstructure, 932
mo_mrm_global_variables::gaugingstation, 786
before
mo_mrm_write_fluxes_states::outputvariable, 885
mo_write_fluxes_states::outputvariable, 890
between
mo_mrm_write_fluxes_states::outputvariable, 886
mo_write_fluxes_states::outputvariable, 891
bias_dp_1d
mo_errormeasures, 152
mo_errormeasures::bias, 768
bias_dp_2d
mo_errormeasures, 152
mo_errormeasures::bias, 769
bias_dp_3d
mo_errormeasures, 152
mo_errormeasures::bias, 769
bias_sp_1d
mo_errormeasures, 152
mo_errormeasures::bias, 769
bias_sp_2d
mo_errormeasures, 153
mo_errormeasures::bias, 769
bias_sp_3d
mo_errormeasures, 153
mo_errormeasures::bias, 769
bulkdens_orgmatter
mo_mpr_constants, 272
bulksurface_resistance
mo_mpr_pet, 295

c1_initstatesm
mo_mhm_constants, 255
mo_mpr_constants, 272
c2tstu
mo_common_mhm_mrm_variables, 109
calc_channel_elevation
mo_mrm_river_head, 414
calc_river_head
mo_mrm_river_head, 415
calc_slope
mo_mrm_river_head, 416
calculate_grid_properties
mo_grid, 192
calculate_l11_flow_accumulation

mo_mrm_net_startup.f90, 996
caldat
mo_julian, 203
caldat360
mo_julian, 205
caldat365
mo_julian, 205
caldatjulian
mo_julian, 206
calendar
mo_julian, 229
calls
mo_mrm_write_fluxes_states::outputvariable, 886
mo_write_fluxes_states::outputvariable, 891
canopy_interc
mo_canopy_interc, 95
canopy_intercept_param
mo_multi_param_reg, 458
cce
mo_sce, 675
cellarea
mo_common_variables::grid, 797
cellcoor
mo_common_variables::grid, 797
celllength
mo_mrm_net_startup, 359
cellsize
mo_common_variables::grid, 798
central_moment_dp
mo_moment, 266
mo_moment::central_moment, 770
central_moment_sp
mo_moment, 266
mo_moment::central_moment, 770
central_moment_var_dp
mo_moment, 267
mo_moment::central_moment_var, 770
central_moment_var_sp
mo_moment, 267
mo_moment::central_moment_var, 770
check
mo_ncread, 468
mo_ncwrite, 489
mo_netcdf, 518
check_consistency_lut_map
mo_read_wrapper, 665
check_optimization_settings
mo_common_mhm_mrm_read_config, 105
chkcst
mo_sce, 676
chunk_config
mo_meteo_forcings, 240
chunk_size
mo_meteo_forcings, 242
circum
mo_template, 708
classified_standard_score_dp
mo_standard_score, 696
mo_standard_score::classified_standard_score, 772
classified_standard_score_sp
mo_standard_score, 696
mo_standard_score::classified_standard_score, 772
clay
mo_mpr_global_variables::soiltype, 914
clock_rate
mo_timer, 721
close
mo_mrm_write_fluxes_states, 447
mo_mrm_write_fluxes_states::outputdataset, 877
mo_netcdf, 518
mo_netcdf::ncdataset, 839
mo_write_fluxes_states, 739
mo_write_fluxes_states::outputdataset, 881
close_netcdf
mo_ncwrite, 490
close_nml
mo_nml, 589
common_check_resolution
mo_common_mhm_mrm_read_config, 106
common_mhm_mrm_read_config
mo_common_mhm_mrm_read_config, 106
common_read_config
mo_common_read_config, 114
comp
mo_sce, 676
compress
mo_string_utils, 701
config_output
mo_mrm_init, 346
constants_init
mo_startup, 697
contact
mo_common_variables, 123
contains
mo_mrm_write_fluxes_states::outputvariable, 886
mo_write_fluxes_states::outputvariable, 891
conventions
mo_common_variables, 123
corr_dp
mo_corr, 142
mo_corr::corr, 772
corr_sp
mo_corr, 142
mo_corr::corr, 773
correlation_dp
mo_moment, 267
mo_moment::correlation, 773
correlation_sp
mo_moment, 267
mo_moment::correlation, 773
cosmic
mo_neutrons, 583
cosmic_alpha
mo_mhm_constants, 255

cosmic_bd
mo_mhm_constants, 255
cosmic_l1
mo_mhm_constants, 255
cosmic_l2
mo_mhm_constants, 255
cosmic_l3
mo_mhm_constants, 255
cosmic_l4
mo_mhm_constants, 256
cosmic_n
mo_mhm_constants, 256
cosmic_vwclat
mo_mhm_constants, 256
count
mo_mrm_write_fluxes_states::outputdataset, 877
mo_mrm_write_fluxes_states::outputvariable, 886
mo_ncwrite::variable, 949
mo_write_fluxes_states::outputdataset, 881
mo_write_fluxes_states::outputvariable, 891
counter
mo_mrm_write_fluxes_states::outputdataset, 878
mo_mrm_write_fluxes_states::outputvariable, 886
mo_write_fluxes_states::outputdataset, 882
mo_write_fluxes_states::outputvariable, 891
covariance_dp
mo_moment, 267
mo_moment::covariance, 774
covariance_sp
mo_moment, 267
mo_moment::covariance, 774
cp0_dp
mo_constants, 132
cp0_sp
mo_constants, 132
cputime
mo_timer, 721
create_netcdf
mo_ncwrite, 492
create_output
mo_mrm_river_head, 416
created
mo_mrm_write_fluxes_states::outputdataset, 878
mo_write_fluxes_states::outputdataset, 882
createoutputfile
mo_mrm_write_fluxes_states, 448
mo_write_fluxes_states, 739
crosscoeffk_dp
mo_corr, 142
mo_corr::crosscoeffk, 774
crosscoeffk_sp
mo_corr, 143
mo_corr::crosscoeffk, 774
crosscorr_dp
mo_corr, 143
mo_corr::crosscorr, 775
crosscorr_sp
mo_corr, 143
mo_corr::crosscorr, 775
cycles1
mo_timer, 721
cycles2
mo_timer, 721
cycles_max
mo_timer, 722

d_ctrper
mo_orderpack, 615
mo_orderpack::ctrper, 776
d_fndnth
mo_orderpack, 615
mo_orderpack::fndnth, 784
d_indmed
mo_orderpack, 615
mo_orderpack::indmed, 803
d_indnth
mo_orderpack, 616
mo_orderpack::indnth, 803
d_inspar
mo_orderpack, 616
mo_orderpack::inspar, 804
d_inssor
mo_orderpack, 616
mo_orderpack::inssor, 805
d_med
mo_orderpack, 616
d_median
mo_orderpack, 617
mo_orderpack::omedian, 875
d_mrgref
mo_orderpack, 617
mo_orderpack::mrgref, 832
d_mrgrnk
mo_orderpack, 617
mo_orderpack::mrgrnk, 833
d_mulcnt
mo_orderpack, 617
mo_orderpack::mulcnt, 835
d_nearless
mo_orderpack, 617
mo_orderpack::nearless, 868
d_rapknr
mo_orderpack, 617
mo_orderpack::rapknr, 902
d_refpar
mo_orderpack, 618
mo_orderpack::refpar, 906
d_refsor
mo_orderpack, 618
mo_orderpack::refsor, 907
mo_orderpack::sort, 919
d_rinpar
mo_orderpack, 618
mo_orderpack::rinpar, 907
d_rnkpar
mo_orderpack, 618
mo_orderpack::rnkpar, 909

d_subsor
mo_orderpack, 618
d_uniinv
mo_orderpack, 619
mo_orderpack::uniinv, 933
d_unipar
mo_orderpack, 619
mo_orderpack::unipar, 934
d_unirnk
mo_orderpack, 619
mo_orderpack::unirnk, 934
d_unista
mo_orderpack, 619
mo_orderpack::unista, 935
d_valmed
mo_orderpack, 619
mo_orderpack::valmed, 939
d_valnth
mo_orderpack, 620
mo_orderpack::valnth, 940
DEPENDENCIES.md, 959
data
mo_mrm_write_fluxes_states::outputvariable, 886
mo_write_fluxes_states::outputvariable, 891
dataset
mo_netcdf::ncdataset, 846
date2dec
mo_julian, 208
date2dec360
mo_julian, 209
date2dec365
mo_julian, 210
date2decjulian
mo_julian, 211
day2mon_average_dp
mo_temporal_aggregation, 710
mo_temporal_aggregation::day2mon_average, 777
day_counter
mo_mrm_write, 446
dayhours
mo_common_constants, 98
mo_constants, 132
daysecs
mo_common_constants, 98
mo_constants, 132
db
mo_mpr_global_variables::soiltype, 914
dbm
mo_mpr_global_variables::soiltype, 914
dchange_dp
mo_anneal, 84
dds
mo_dds, 147
dds_r
mo_common_mhm_mrm_variables, 109
dec2date
mo_julian, 212
dec2date360
mo_julian, 214
dec2date365
mo_julian, 216
dec2datejulian
mo_julian, 217
deg2rad_dp
mo_constants, 132
deg2rad_sp
mo_constants, 133
deltah
mo_mrm_constants, 324
dend
mo_common_variables::period, 901
depth
mo_mpr_global_variables::soiltype, 914
desilets_a0
mo_mhm_constants, 256
desilets_a1
mo_mhm_constants, 256
desilets_a2
mo_mhm_constants, 256
desiletsn0
mo_neutrons, 584
dimension
mo_netcdf::ncdimension, 866
dimid
mo_ncwrite::dims, 778
dimids
mo_ncwrite::variable, 949
dimtypes
mo_ncwrite::variable, 949
dirabsvappressure
mo_global_variables, 181
dirbankfullrunoff
mo_mrm_global_variables, 335
dircommonfiles
mo_common_variables, 124
dirconfigout
mo_common_variables, 124
direvapotranspiration
mo_global_variables, 181
dirgauges
mo_mrm_global_variables, 335
dirgridded_lai
mo_mpr_global_variables, 287
dirlcover
mo_common_variables, 124
dirmaxtemperature
mo_global_variables, 181
dirmintemperature
mo_global_variables, 181
dirmorpho
mo_common_variables, 124
dirnetradiation
mo_global_variables, 181
dirneutrons
mo_global_variables, 181

dirout
mo_common_variables, 124
dirprecipitation
mo_global_variables, 182
dirreferenceet
mo_global_variables, 182
dirrestartin
mo_common_mhm_mrm_variables, 110
dirrestartout
mo_common_variables, 124
dirsoil_moisture
mo_global_variables, 182
dirtemperature
mo_global_variables, 182
dirtotalrunoff
mo_mrm_global_variables, 335
dirwindspeed
mo_global_variables, 182
divide_string
mo_string_utils, 702
dnc
mo_ncwrite, 513
dp
mo_kind, 230
dp2str
mo_string_utils, 703
mo_string_utils::num2str, 873
dpc
mo_kind, 230
dstart
mo_common_variables::period, 901
duffiedelta1
mo_mhm_constants, 256
duffiedelta2
mo_mhm_constants, 257
duffiedr
mo_mhm_constants, 257
dummy_mpr, 83
dummy_mrm, 83
dump_netcdf_1d_dp
mo_ncwrite, 493
mo_ncwrite::dump_netcdf, 778
dump_netcdf_1d_i4
mo_ncwrite, 493
mo_ncwrite::dump_netcdf, 779
dump_netcdf_1d_sp
mo_ncwrite, 494
mo_ncwrite::dump_netcdf, 779
dump_netcdf_2d_dp
mo_ncwrite, 494
mo_ncwrite::dump_netcdf, 779
dump_netcdf_2d_i4
mo_ncwrite, 495
mo_ncwrite::dump_netcdf, 779
dump_netcdf_2d_sp
mo_ncwrite, 495
mo_ncwrite::dump_netcdf, 780
dump_netcdf_3d_dp
mo_ncwrite, 496
mo_ncwrite::dump_netcdf, 780
dump_netcdf_3d_i4
mo_ncwrite, 496
mo_ncwrite::dump_netcdf, 780
dump_netcdf_3d_sp
mo_ncwrite, 497
mo_ncwrite::dump_netcdf, 780
dump_netcdf_4d_dp
mo_ncwrite, 497
mo_ncwrite::dump_netcdf, 780
dump_netcdf_4d_i4
mo_ncwrite, 498
mo_ncwrite::dump_netcdf, 781
dump_netcdf_4d_sp
mo_ncwrite, 498
mo_ncwrite::dump_netcdf, 781
dump_netcdf_5d_dp
mo_ncwrite, 499
mo_ncwrite::dump_netcdf, 781
dump_netcdf_5d_i4
mo_ncwrite, 499
mo_ncwrite::dump_netcdf, 781
dump_netcdf_5d_sp
mo_ncwrite, 500
mo_ncwrite::dump_netcdf, 781

eps_dp
mo_common_constants, 98
eps_sp
mo_common_constants, 98
equal_dp
mo_utils, 730
mo_utils::eq, 782
mo_utils::equal, 783
equal_sp
mo_utils, 730
mo_utils::eq, 782
mo_utils::equal, 783
equalncdimensions
mo_netcdf, 519
equalstrings
mo_string_utils, 703
euler
mo_constants, 133
euler_d
mo_constants, 133
eval_interface
mo_optimization_utils::eval_interface, 784
evalper
mo_common_mhm_mrm_variables, 110
evap_coeff
mo_global_variables, 182
extract_basin_avg_tws
mo_objective_function, 595
extract_runoff
mo_mrm_objective_function_runoff, 375
extraterr_rad_approx
mo_pet, 633

fday_pet
mo_global_variables, 182
fday_prec
mo_global_variables, 183
fday_temp
mo_global_variables, 183
feddes_et_reduction
mo_soil_moisture, 688
field_cap
mo_mpr_soilmoist, 312
field_cap_c1
mo_mpr_constants, 272
field_cap_c2
mo_mpr_constants, 272
file
mo_netcdf::ncdataset, 846
file_aspect
mo_mpr_file, 281
file_config
mo_common_file, 101
mo_mrm_file, 329
file_daily_discharge
mo_mrm_file, 329
file_defoutput
mo_file, 175
mo_mrm_file, 329
file_dem
mo_common_file, 101
file_facc
mo_mrm_file, 329
file_fdir
mo_mrm_file, 329
file_gaugeloc
mo_mrm_file, 330
file_geolut
mo_mpr_file, 281
file_gw_output
mo_mrm_file, 330
file_hydrogeoclass
mo_mpr_file, 281
file_laiclass
mo_mpr_file, 281
file_lailut
mo_mpr_file, 282
file_main
mo_file, 176
mo_mpr_file, 282
mo_mrm_file, 330
file_meteo_binary_end
mo_mpr_file, 282
file_meteo_header
mo_mpr_file, 282
file_mrm_output
mo_mrm_file, 330
file_namelist_mhm
mo_file, 176
file_namelist_mhm_param
mo_file, 176
file_namelist_mpr
mo_mpr_file, 282
file_namelist_mpr_param
mo_mpr_file, 282
file_namelist_mrm
mo_mrm_file, 330
file_namelist_param_mrm
mo_mrm_file, 330
file_opti
mo_common_mhm_mrm_file, 103
file_opti_nml
mo_common_mhm_mrm_file, 104
file_slope
mo_mpr_file, 282
mo_mrm_file, 330
file_soil_database
mo_mpr_file, 283
file_soil_database_1
mo_mpr_file, 283
file_soilclass
mo_mpr_file, 283
filelatlon
mo_common_variables, 125
filename
mo_netcdf::ncdataset, 846
filenametotalrunoff
mo_mrm_global_variables, 335
filetws
mo_global_variables, 183
finish
mo_finish, 177
flowdurationcurve
mo_mrm_signatures, 428
fluxesunit
mo_write_fluxes_states, 740
fname
mo_global_variables::twsstructure, 932
mo_mrm_global_variables::gaugingstation, 786
mo_netcdf::ncdataset, 846
fnight_pet
mo_global_variables, 183
fnight_prec
mo_global_variables, 183
fnight_temp
mo_global_variables, 183
four1_dp
mo_corr, 143
mo_corr::four1, 785
four1_sp
mo_corr, 143
mo_corr::four1, 785
fourrow_dp
mo_corr, 143
mo_corr::fourrow, 785
fourrow_sp
mo_corr, 144
mo_corr::fourrow, 785
fracsealed_cityarea

mo_mpr_global_variables, 287
g0_b
mo_ncwrite::variable, 950
g0_d
mo_ncwrite::variable, 950
g0_f
mo_ncwrite::variable, 950
g0_i
mo_ncwrite::variable, 950
g1_b
mo_ncwrite::variable, 950
g1_d
mo_ncwrite::variable, 950
g1_f
mo_ncwrite::variable, 950
g1_i
mo_ncwrite::variable, 950
g2_b
mo_ncwrite::variable, 950
g2_d
mo_ncwrite::variable, 951
g2_f
mo_ncwrite::variable, 951
g2_i
mo_ncwrite::variable, 951
g3_b
mo_ncwrite::variable, 951
g3_d
mo_ncwrite::variable, 951
g3_f
mo_ncwrite::variable, 951
g3_i
mo_ncwrite::variable, 951
g4_b
mo_ncwrite::variable, 951
g4_d
mo_ncwrite::variable, 951
g4_f
mo_ncwrite::variable, 952
g4_i
mo_ncwrite::variable, 952
gatt
mo_ncwrite, 513
gauge
mo_mrm_global_variables, 335
gaugeid
mo_mrm_global_variables::gaugingstation, 786
gaugeidlist
mo_mrm_global_variables::basininfo_mrm, 767
gaugeindexlist
mo_mrm_global_variables::basininfo_mrm, 767
gaugenodelist
mo_mrm_global_variables::basininfo_mrm, 767
generate_neighborhood_weight_dp
mo_anneal, 85
mo_anneal::generate_neighborhood_weight, 787
generate_soil_database
mo_soil_database, 686
generatenewparameterset_dp
mo_mcmc, 235
genuchten
mo_mpr_soilmoist, 313
geocoordinates
mo_grid, 194
geounitkar
mo_mpr_global_variables, 287
geounitlist
mo_mpr_global_variables, 287
get_distance_two_lat_lon_points
mo_mrm_net_startup, 360
get_info
mo_ncread, 469
get_ncdim
mo_ncread, 471
get_ncdimatt
mo_ncread, 473
get_ncvar_0d_dp
mo_ncread, 474
mo_ncread::get_ncvar, 788
get_ncvar_0d_i1
mo_ncread, 474
mo_ncread::get_ncvar, 788
get_ncvar_0d_i4
mo_ncread, 475
mo_ncread::get_ncvar, 789
get_ncvar_0d_sp
mo_ncread, 475
mo_ncread::get_ncvar, 789
get_ncvar_1d_dp
mo_ncread, 476
mo_ncread::get_ncvar, 789
get_ncvar_1d_i1
mo_ncread, 476
mo_ncread::get_ncvar, 789
get_ncvar_1d_i4
mo_ncread, 477
mo_ncread::get_ncvar, 789
get_ncvar_1d_sp
mo_ncread, 477
mo_ncread::get_ncvar, 790
get_ncvar_2d_dp
mo_ncread, 478
mo_ncread::get_ncvar, 790
get_ncvar_2d_i1
mo_ncread, 478
mo_ncread::get_ncvar, 790
get_ncvar_2d_i4
mo_ncread, 479
mo_ncread::get_ncvar, 790
get_ncvar_2d_sp
mo_ncread, 479
mo_ncread::get_ncvar, 790
get_ncvar_3d_dp
mo_ncread, 480
mo_ncread::get_ncvar, 791
get_ncvar_3d_i1

mo_ncread, 480
mo_ncread::get_ncvar, 791
get_ncvar_3d_i4
mo_ncread, 481
mo_ncread::get_ncvar, 791
get_ncvar_3d_sp
mo_ncread, 481
mo_ncread::get_ncvar, 791
get_ncvar_4d_dp
mo_ncread, 482
mo_ncread::get_ncvar, 792
get_ncvar_4d_i1
mo_ncread, 482
mo_ncread::get_ncvar, 792
get_ncvar_4d_i4
mo_ncread, 483
mo_ncread::get_ncvar, 792
get_ncvar_4d_sp
mo_ncread, 483
mo_ncread::get_ncvar, 792
get_ncvar_5d_dp
mo_ncread, 484
mo_ncread::get_ncvar, 792
get_ncvar_5d_i1
mo_ncread, 484
mo_ncread::get_ncvar, 793
get_ncvar_5d_i4
mo_ncread, 485
mo_ncread::get_ncvar, 793
get_ncvar_5d_sp
mo_ncread, 485
mo_ncread::get_ncvar, 793
get_ncvaratt
mo_ncread, 486
get_time_vector_and_select
mo_read_forcing_nc, 644
get_timeseed_i4_0d
mo_xor4096, 748
mo_xor4096::get_timeseed, 793
get_timeseed_i4_1d
mo_xor4096, 748
mo_xor4096::get_timeseed, 794
get_timeseed_i8_0d
mo_xor4096, 748
mo_xor4096::get_timeseed, 794
get_timeseed_i8_1d
mo_xor4096, 748
mo_xor4096::get_timeseed, 794
getattribute
mo_netcdf::ncdataset, 839
mo_netcdf::ncdimension, 851
getdata
mo_netcdf::ncdimension, 852
getdata1df32
mo_netcdf, 519
mo_netcdf::ncdimension, 852
getdata1df64
mo_netcdf, 519
mo_netcdf::ncdimension, 852
getdata1di16
mo_netcdf, 520
mo_netcdf::ncdimension, 852
getdata1di32
mo_netcdf, 520
mo_netcdf::ncdimension, 852
getdata1di64
mo_netcdf, 521
mo_netcdf::ncdimension, 852
getdata1di8
mo_netcdf, 521
mo_netcdf::ncdimension, 853
getdata2df32
mo_netcdf, 522
mo_netcdf::ncdimension, 853
getdata2df64
mo_netcdf, 522
mo_netcdf::ncdimension, 853
getdata2di16
mo_netcdf, 523
mo_netcdf::ncdimension, 853
getdata2di32
mo_netcdf, 523
mo_netcdf::ncdimension, 853
getdata2di64
mo_netcdf, 524
mo_netcdf::ncdimension, 853
getdata2di8
mo_netcdf, 524
mo_netcdf::ncdimension, 853
getdata3df32
mo_netcdf, 525
mo_netcdf::ncdimension, 853
getdata3df64
mo_netcdf, 525
mo_netcdf::ncdimension, 853
getdata3di16
mo_netcdf, 526
mo_netcdf::ncdimension, 854
getdata3di32
mo_netcdf, 526
mo_netcdf::ncdimension, 854
getdata3di64
mo_netcdf, 527
mo_netcdf::ncdimension, 854
getdata3di8
mo_netcdf, 527
mo_netcdf::ncdimension, 854
getdata4df32
mo_netcdf, 528
mo_netcdf::ncdimension, 854
getdata4df64
mo_netcdf, 528
mo_netcdf::ncdimension, 854
getdata4di16
mo_netcdf, 529
mo_netcdf::ncdimension, 854

getdata4di32
mo_netcdf, 529
mo_netcdf::ncdimension, 854
getdata4di64
mo_netcdf, 530
mo_netcdf::ncdimension, 854
getdata4di8
mo_netcdf, 530
mo_netcdf::ncdimension, 855
getdata5df32
mo_netcdf, 531
mo_netcdf::ncdimension, 855
getdata5df64
mo_netcdf, 531
mo_netcdf::ncdimension, 855
getdata5di16
mo_netcdf, 532
mo_netcdf::ncdimension, 855
getdata5di32
mo_netcdf, 532
mo_netcdf::ncdimension, 855
getdata5di64
mo_netcdf, 533
mo_netcdf::ncdimension, 855
getdata5di8
mo_netcdf, 533
mo_netcdf::ncdimension, 855
getdatascalarf32
mo_netcdf, 534
mo_netcdf::ncdimension, 855
getdatascalarf64
mo_netcdf, 534
mo_netcdf::ncdimension, 855
getdatascalari16
mo_netcdf, 535
mo_netcdf::ncdimension, 856
getdatascalari32
mo_netcdf, 535
mo_netcdf::ncdimension, 856
getdatascalari64
mo_netcdf, 536
mo_netcdf::ncdimension, 856
getdatascalari8
mo_netcdf, 536
mo_netcdf::ncdimension, 856
getdimension
mo_netcdf::ncdataset, 839
getdimensionbyid
mo_netcdf, 537
mo_netcdf::ncdataset, 840
getdimensionbyname
mo_netcdf, 537
mo_netcdf::ncdataset, 840
getdimensionlength
mo_netcdf, 537
getdimensionname
mo_netcdf, 538
getdimensions
mo_netcdf::ncdimension, 856
getdtype
mo_netcdf::ncdimension, 856
getdtypefrominteger
mo_netcdf, 538
getdtypefromstring
mo_netcdf, 539
getfillvalue
mo_netcdf::ncdimension, 856
getglobalattributechar
mo_netcdf, 539
mo_netcdf::ncdataset, 840
getglobalattributef32
mo_netcdf, 539
mo_netcdf::ncdataset, 840
getglobalattributef64
mo_netcdf, 540
mo_netcdf::ncdataset, 840
getglobalattributei16
mo_netcdf, 540
mo_netcdf::ncdataset, 840
getglobalattributei32
mo_netcdf, 541
mo_netcdf::ncdataset, 840
getglobalattributei64
mo_netcdf, 541
mo_netcdf::ncdataset, 840
getglobalattributei8
mo_netcdf, 542
mo_netcdf::ncdataset, 841
getname
mo_netcdf::ncdimension, 857
getnodimensions
mo_netcdf, 542
mo_netcdf::ncdimension, 857
getnovariables
mo_netcdf, 542
mo_netcdf::ncdataset, 841
getpnt
mo_sce, 677
getreaddatashape
mo_netcdf, 543
getshape
mo_netcdf::ncdimension, 857
gettemperature_dp
mo_anneal, 85
mo_anneal::gettemperature, 795
getunlimiteddimension
mo_netcdf, 544
mo_netcdf::ncdataset, 841
getvariable
mo_netcdf::ncdataset, 841
getvariableattributechar
mo_netcdf, 545
mo_netcdf::ncdimension, 857
getvariableattributef32
mo_netcdf, 545
mo_netcdf::ncdimension, 857

getvariableattributef64
mo_netcdf, 546
mo_netcdf::ncdimension, 857
getvariableattributei16
mo_netcdf, 546
mo_netcdf::ncdimension, 857
getvariableattributei32
mo_netcdf, 546
mo_netcdf::ncdimension, 857
getvariableattributei64
mo_netcdf, 547
mo_netcdf::ncdimension, 858
getvariableattributei8
mo_netcdf, 547
mo_netcdf::ncdimension, 858
getvariablebyname
mo_netcdf, 548
mo_netcdf::ncdataset, 842
getvariabledimensions
mo_netcdf, 548
getvariabledtype
mo_netcdf, 548
getvariablefillvaluef32
mo_netcdf, 549
mo_netcdf::ncdimension, 858
getvariablefillvaluef64
mo_netcdf, 549
mo_netcdf::ncdimension, 858
getvariablefillvaluei16
mo_netcdf, 549
mo_netcdf::ncdimension, 858
getvariablefillvaluei32
mo_netcdf, 549
mo_netcdf::ncdimension, 858
getvariablefillvaluei64
mo_netcdf, 549
mo_netcdf::ncdimension, 858
getvariablefillvaluei8
mo_netcdf, 549
mo_netcdf::ncdimension, 858
getvariableids
mo_netcdf, 550
mo_netcdf::ncdataset, 842
getvariablename
mo_netcdf, 550
getvariables
mo_netcdf, 550
mo_netcdf::ncdataset, 842
getvariableshape
mo_netcdf, 550
given_ts
mo_mrm_constants, 324
global_parameters
mo_common_variables, 125
global_parameters_name
mo_common_variables, 125
gravity_dp
mo_constants, 133
gravity_sp
mo_constants, 133
greaterequal_dp
mo_utils, 730
mo_utils::ge, 787
mo_utils::greaterequal, 796
greaterequal_sp
mo_utils, 730
mo_utils::ge, 787
mo_utils::greaterequal, 796
gw_coupling
mo_mrm_global_variables, 335

h2odens
mo_mhm_constants, 257
harsamconst
mo_mhm_constants, 257
hasattribute
mo_netcdf, 551
mo_netcdf::ncdimension, 858
hasdimension
mo_netcdf, 551
mo_netcdf::ncdataset, 842
hasvariable
mo_netcdf, 551
mo_netcdf::ncdataset, 842
high_res_grid
mo_common_variables::gridremapper, 800
history
mo_common_variables, 125
horizondepth_mhm
mo_mpr_global_variables, 287
hour2day_average_dp
mo_temporal_aggregation, 711
mo_temporal_aggregation::hour2day_average, 802
hoursecs
mo_common_constants, 98
hydro_cond
mo_mpr_soilmoist, 314

i1
mo_kind, 230
i2
mo_kind, 230
i4
mo_kind, 231
i42str
mo_string_utils, 703
mo_string_utils::num2str, 873
i4array2str
mo_string_utils, 704
mo_string_utils::numarray2str, 874
i8
mo_kind, 231
i82str
mo_string_utils, 704
mo_string_utils::num2str, 873
i_ctrper

mo_orderpack, 620
mo_orderpack::ctrper, 776
i_fndnth
mo_orderpack, 620
mo_orderpack::fndnth, 784
i_indmed
mo_orderpack, 620
mo_orderpack::indmed, 803
i_indnth
mo_orderpack, 620
mo_orderpack::indnth, 803
i_inspar
mo_orderpack, 621
mo_orderpack::inspar, 804
i_inssor
mo_orderpack, 621
mo_orderpack::inssor, 805
i_med
mo_orderpack, 621
i_median
mo_orderpack, 621
mo_orderpack::omedian, 875
i_mrgref
mo_orderpack, 622
mo_orderpack::mrgref, 832
i_mrgrnk
mo_orderpack, 622
mo_orderpack::mrgrnk, 833
i_mulcnt
mo_orderpack, 622
mo_orderpack::mulcnt, 835
i_nearless
mo_orderpack, 622
mo_orderpack::nearless, 868
i_rapknr
mo_orderpack, 622
mo_orderpack::rapknr, 903
i_refpar
mo_orderpack, 622
mo_orderpack::refpar, 906
i_refsor
mo_orderpack, 622
mo_orderpack::refsor, 907
mo_orderpack::sort, 919
i_rinpar
mo_orderpack, 623
mo_orderpack::rinpar, 908
i_rnkpar
mo_orderpack, 623
mo_orderpack::rnkpar, 910
i_subsor
mo_orderpack, 623
i_uniinv
mo_orderpack, 624
mo_orderpack::uniinv, 933
i_unipar
mo_orderpack, 624
mo_orderpack::unipar, 934
i_unirnk
mo_orderpack, 624
mo_orderpack::unirnk, 935
i_unista
mo_orderpack, 624
mo_orderpack::unista, 935
i_valmed
mo_orderpack, 624
mo_orderpack::valmed, 939
i_valnth
mo_orderpack, 624
mo_orderpack::valnth, 940
ibasin
mo_mrm_write_fluxes_states::outputdataset, 878
mo_write_fluxes_states::outputdataset, 882
id
mo_common_variables::grid, 798
mo_mpr_global_variables::soiltype, 914
mo_mrm_write_fluxes_states::outputdataset, 878
mo_netcdf::ncdataset, 846
mo_netcdf::ncdimension, 866
mo_write_fluxes_states::outputdataset, 882
idont
mo_orderpack, 630
iend
mo_common_variables::grid, 798
iflag_cordinate_sys
mo_common_variables, 125
iflag_soildb
mo_mpr_global_variables, 287
in_bound
mo_common_functions, 102
inflowgauge
mo_mrm_global_variables, 336
inflowgaugeheadwater
mo_mrm_global_variables::basininfo_mrm, 767
inflowgaugeidlist
mo_mrm_global_variables::basininfo_mrm, 767
inflowgaugeindexlist
mo_mrm_global_variables::basininfo_mrm, 767
inflowgaugenodelist
mo_mrm_global_variables::basininfo_mrm, 767
init_eff_params
mo_mpr_startup, 319
init_lowres_level
mo_grid, 194
init_masked_zeros_l0
mo_mrm_river_head, 417
initncdataset
mo_netcdf, 551
mo_netcdf::ncdataset, 843
initncdimension
mo_netcdf, 551
initncvariable
mo_netcdf, 552
mo_netcdf::ncdimension, 859
inputformat_gridded_lai
mo_mpr_global_variables, 288

inputformat_meteo_forcings
mo_global_variables, 183
intgrandfast
mo_neutrons, 585
iper_thres_runoff
mo_multi_param_reg, 459
irow
mo_kind::sprs2_dp, 924
mo_kind::sprs2_sp, 926
is_finite_dp
mo_utils, 731
mo_utils::is_finite, 806
is_finite_sp
mo_utils, 731
mo_utils::is_finite, 806
is_nan_dp
mo_utils, 731
mo_utils::is_nan, 806
is_nan_sp
mo_utils, 731
mo_utils::is_nan, 806
is_normal_dp
mo_utils, 731
mo_utils::is_normal, 807
is_normal_sp
mo_utils, 731
mo_utils::is_normal, 807
is_present
mo_mpr_global_variables::soiltype, 914
is_read
mo_meteo_forcings, 243
is_start
mo_mrm_global_variables, 336
isdatasetunlimited
mo_netcdf, 552
istart
mo_common_variables::grid, 798
isunlimited
mo_netcdf::ncdataset, 843
mo_netcdf::ncdimension, 859
isunlimiteddimension
mo_netcdf, 552
isunlimitedvariable
mo_netcdf, 552
itest
mo_template, 709

jarvis_et_reduction
mo_soil_moisture, 689
jcol
mo_kind::sprs2_dp, 924
mo_kind::sprs2_sp, 926
julday
mo_julian, 219
julday360
mo_julian, 220
julday365
mo_julian, 221
juldayjulian
mo_julian, 223
julend
mo_common_variables::period, 901
julstart
mo_common_variables::period, 901

karman
mo_mpr_constants, 272
karstic_layer
mo_multi_param_reg, 460
kge_dp_1d
mo_errormeasures, 153
mo_errormeasures::kge, 808
kge_dp_2d
mo_errormeasures, 153
mo_errormeasures::kge, 808
kge_dp_3d
mo_errormeasures, 153
mo_errormeasures::kge, 809
kge_sp_1d
mo_errormeasures, 153
mo_errormeasures::kge, 809
kge_sp_2d
mo_errormeasures, 154
mo_errormeasures::kge, 809
kge_sp_3d
mo_errormeasures, 154
mo_errormeasures::kge, 809
kgenocorr_dp_1d
mo_errormeasures, 154
mo_errormeasures::kgenocorr, 810
kgenocorr_dp_2d
mo_errormeasures, 154
mo_errormeasures::kgenocorr, 810
kgenocorr_dp_3d
mo_errormeasures, 154
mo_errormeasures::kgenocorr, 811
kgenocorr_sp_1d
mo_errormeasures, 154
mo_errormeasures::kgenocorr, 811
kgenocorr_sp_2d
mo_errormeasures, 155
mo_errormeasures::kgenocorr, 811
kgenocorr_sp_3d
mo_errormeasures, 155
mo_errormeasures::kgenocorr, 811
ks
mo_mpr_global_variables::soiltype, 915
ks_c
mo_mpr_constants, 272
kurtosis_dp
mo_moment, 268
mo_moment::kurtosis, 812
kurtosis_sp
mo_moment, 268
mo_moment::kurtosis, 812

l0_asp
mo_mpr_global_variables, 288

l0_basin
mo_common_variables, 125
l0_celerity
mo_mrm_global_variables, 336
l0_channel_depth
mo_mrm_global_variables, 336
l0_channel_elevation
mo_mrm_global_variables, 336
l0_check_input
mo_mpr_startup, 319
l0_check_input_routing
mo_mrm_init, 347
l0_coloutlet
mo_mrm_global_variables::basininfo_mrm, 768
l0_dracell
mo_mrm_global_variables, 336
l0_drasc
mo_mrm_global_variables, 336
l0_elev
mo_common_variables, 126
l0_facc
mo_mrm_global_variables, 337
l0_fdir
mo_mrm_global_variables, 337
l0_floodplain
mo_mrm_global_variables, 337
l0_fractionalcover_in_lx
mo_upscaling_operators, 723
l0_gaugeloc
mo_mrm_global_variables, 337
l0_geounit
mo_mpr_global_variables, 288
l0_grid_setup
mo_grid, 195
l0_gridded_lai
mo_mpr_global_variables, 288
l0_inflowgaugeloc
mo_mrm_global_variables, 337
l0_l11_remap
mo_common_variables, 126
mo_mrm_global_variables, 337
l0_l1_remap
mo_common_variables, 126
l0_lcover
mo_common_variables, 126
l0_noutlet
mo_mrm_global_variables, 338
mo_mrm_global_variables::basininfo_mrm, 768
l0_river_head_mon_sum
mo_mrm_global_variables, 338
l0_rowoutlet
mo_mrm_global_variables::basininfo_mrm, 768
l0_slope
mo_mpr_global_variables, 288
mo_mrm_global_variables, 338
l0_slope_emp
mo_mpr_global_variables, 288
l0_soilid
mo_mpr_global_variables, 289
l0_streamnet
mo_mrm_global_variables, 338
l0_variable_init
mo_mpr_startup, 320
l11_afloodplain
mo_mrm_global_variables, 338
l11_areacell
mo_mrm_global_variables, 338
l11_bankfull_runoff_in
mo_mrm_global_variables, 338
l11_basin
mo_common_variables, 126
l11_c1
mo_mrm_global_variables, 339
l11_c2
mo_mrm_global_variables, 339
l11_calc_celerity
mo_mrm_net_startup, 361
l11_celerity
mo_mrm_global_variables, 339
l11_cellcoor
mo_mrm_global_variables, 339
l11_colout
mo_mrm_global_variables, 339
l11_facc
mo_mrm_global_variables, 339
l11_fcol
mo_mrm_global_variables, 339
l11_fdir
mo_mrm_global_variables, 340
l11_flow_accumulation
mo_mrm_net_startup, 362
l11_flow_direction
mo_mrm_net_startup, 363
l11_fraction_sealed_floodplain
mo_mrm_net_startup, 365
l11_fromn
mo_mrm_global_variables, 340
l11_frow
mo_mrm_global_variables, 340
l11_k
mo_mrm_global_variables, 340
l11_l1_id
mo_mrm_global_variables, 340
l11_l1_mapping
mo_mrm_net_startup, 366
l11_label
mo_mrm_global_variables, 340
l11_length
mo_mrm_global_variables, 341
l11_link_location
mo_mrm_net_startup, 367
l11_linkin_facc
mo_mrm_global_variables, 341
l11_meandering
mo_mrm_global_variables, 341
l11_netperm

mo_mrm_global_variables, 341
l11_nlinkfracfpimp
mo_mrm_global_variables, 341
l11_noutlets
mo_mrm_global_variables, 341
l11_qmod
mo_mrm_global_variables, 341
l11_qout
mo_mrm_global_variables, 342
l11_qtin
mo_mrm_global_variables, 342
l11_qtr
mo_mrm_global_variables, 342
l11_rorder
mo_mrm_global_variables, 342
l11_routing
mo_mrm_routing, 421
l11_routing_order
mo_mrm_net_startup, 368
l11_rowout
mo_mrm_global_variables, 342
l11_runoff_acc
mo_mrm_routing, 423
l11_set_drain_outlet_gauges
mo_mrm_net_startup, 369
l11_set_network_topology
mo_mrm_net_startup, 370
l11_sink
mo_mrm_global_variables, 342
l11_slope
mo_mrm_global_variables, 343
l11_stream_features
mo_mrm_net_startup, 371
l11_tcol
mo_mrm_global_variables, 343
l11_ton
mo_mrm_global_variables, 343
l11_trow
mo_mrm_global_variables, 343
l11_tsrout
mo_mrm_global_variables, 343
l11_xi
mo_mrm_global_variables, 343
l1_absvappress
mo_global_variables, 184
l1_aeroresist
mo_mpr_global_variables, 289
l1_aetcanopy
mo_global_variables, 184
l1_aetsealed
mo_global_variables, 184
l1_aetsoil
mo_global_variables, 184
l1_alpha
mo_mpr_global_variables, 289
l1_baseflow
mo_global_variables, 184
l1_degday
mo_mpr_global_variables, 289
l1_degdayinc
mo_mpr_global_variables, 289
l1_degdaymax
mo_mpr_global_variables, 289
l1_degdaynopre
mo_mpr_global_variables, 289
l1_et
mo_global_variables, 184
l1_et_mask
mo_global_variables, 184
l1_fasp
mo_mpr_global_variables, 290
l1_fastrunoff
mo_global_variables, 185
l1_froots
mo_mpr_global_variables, 290
l1_fsealed
mo_mpr_global_variables, 290
l1_harsamcoeff
mo_mpr_global_variables, 290
l1_infilsoil
mo_global_variables, 185
l1_inter
mo_global_variables, 185
l1_jarvis_thresh_c1
mo_mpr_global_variables, 290
l1_karstloss
mo_mpr_global_variables, 290
l1_kbaseflow
mo_mpr_global_variables, 290
l1_kfastflow
mo_mpr_global_variables, 291
l1_kperco
mo_mpr_global_variables, 291
l1_kslowflow
mo_mpr_global_variables, 291
l1_l11_id
mo_mrm_global_variables, 343
l1_l11_remap
mo_common_variables, 126
mo_mrm_global_variables, 344
l1_maxinter
mo_mpr_global_variables, 291
l1_melt
mo_global_variables, 185
l1_netrad
mo_global_variables, 185
l1_neutrons
mo_global_variables, 185
l1_neutronsdata
mo_global_variables, 185
l1_neutronsdata_mask
mo_global_variables, 186
l1_percol
mo_global_variables, 186
l1_pet
mo_global_variables, 186

l1_pet_calc
mo_global_variables, 186
l1_pet_weights
mo_global_variables, 186
l1_petlaicorfactor
mo_mpr_global_variables, 291
l1_pre
mo_global_variables, 186
l1_pre_weights
mo_global_variables, 186
l1_preeffect
mo_global_variables, 187
l1_prietayalpha
mo_mpr_global_variables, 291
l1_rain
mo_global_variables, 187
l1_runoffseal
mo_global_variables, 187
l1_satstw
mo_global_variables, 187
l1_sealedthresh
mo_mpr_global_variables, 292
l1_sealstw
mo_global_variables, 187
l1_slowrunoff
mo_global_variables, 187
l1_sm
mo_global_variables, 187
l1_sm_mask
mo_global_variables, 188
l1_snow
mo_global_variables, 188
l1_snowpack
mo_global_variables, 188
l1_soilmoist
mo_global_variables, 188
l1_soilmoistexp
mo_mpr_global_variables, 292
l1_soilmoistfc
mo_mpr_global_variables, 292
l1_soilmoistsat
mo_mpr_global_variables, 292
l1_surfresist
mo_mpr_global_variables, 292
l1_temp
mo_global_variables, 188
l1_temp_weights
mo_global_variables, 188
l1_tempthresh
mo_mpr_global_variables, 292
l1_throughfall
mo_global_variables, 189
l1_tmax
mo_global_variables, 189
l1_tmin
mo_global_variables, 189
l1_total_runoff
mo_global_variables, 189
mo_runoff, 672
l1_total_runoff_in
mo_mrm_global_variables, 344
l1_unsatstw
mo_global_variables, 189
l1_unsatthresh
mo_mpr_global_variables, 292
l1_wiltingpoint
mo_mpr_global_variables, 293
l1_windspeed
mo_global_variables, 189
l2_variable_init
mo_startup, 698
lai_factor_surfresi
mo_mpr_constants, 272
lai_offset_surfresi
mo_mpr_constants, 273
lailut
mo_mpr_global_variables, 293
laiper
mo_mpr_global_variables, 293
laiunitlist
mo_mpr_global_variables, 293
lc_year_end
mo_common_variables, 126
lc_year_start
mo_common_variables, 127
lcfilename
mo_common_variables, 127
lcyearid
mo_common_mhm_mrm_variables, 110
ld
mo_mpr_global_variables::soiltype, 915
left_bound
mo_common_variables::gridremapper, 801
len
mo_kind::sprs2_dp, 925
mo_kind::sprs2_sp, 926
mo_ncwrite::dims, 778
length_error
mo_nml, 593
lesserequal_dp
mo_utils, 731
mo_utils::le, 812
mo_utils::lesserequal, 813
lesserequal_sp
mo_utils, 731
mo_utils::le, 812
mo_utils::lesserequal, 813
level0
mo_common_variables, 127
level1
mo_common_variables, 127
level11
mo_mrm_global_variables, 344
level2
mo_global_variables, 189
lgt

mo_kind, 231
limb_densities
mo_mrm_signatures, 430
linfit_dp
mo_linfit, 232
mo_linfit::linfit, 814
linfit_sp
mo_linfit, 233
mo_linfit::linfit, 814
lnnse_dp_1d
mo_errormeasures, 155
mo_errormeasures::lnnse, 815
lnnse_dp_2d
mo_errormeasures, 155
mo_errormeasures::lnnse, 815
lnnse_dp_3d
mo_errormeasures, 155
mo_errormeasures::lnnse, 815
lnnse_sp_1d
mo_errormeasures, 155
mo_errormeasures::lnnse, 815
lnnse_sp_2d
mo_errormeasures, 155
mo_errormeasures::lnnse, 815
lnnse_sp_3d
mo_errormeasures, 156
mo_errormeasures::lnnse, 815
locate_0d_dp
mo_utils, 732
mo_utils::locate, 816
locate_0d_sp
mo_utils, 732
mo_utils::locate, 816
locate_1d_dp
mo_utils, 732
mo_utils::locate, 817
locate_1d_sp
mo_utils, 732
mo_utils::locate, 817
log2str
mo_string_utils, 704
mo_string_utils::num2str, 873
loglikelihood_evin2013_2
mo_mrm_objective_function_runoff, 377
loglikelihood_stddev
mo_mrm_objective_function_runoff, 378
loglikelihood_trend_no_autocorr
mo_mrm_objective_function_runoff, 380
lookupintegral
mo_neutrons, 586
low_res_grid
mo_common_variables::gridremapper, 801
lower_bound
mo_common_variables::gridremapper, 801
lowres_id_on_highres
mo_common_variables::gridremapper, 801

mad_dp
mo_mad, 233
mo_mad::mad, 817
mad_sp
mo_mad, 233
mo_mad::mad, 817
mad_val_dp
mo_mad, 234
mo_mad::mad, 818
mad_val_sp
mo_mad, 234
mo_mad::mad, 818
mae_dp_1d
mo_errormeasures, 156
mo_errormeasures::mae, 818
mae_dp_2d
mo_errormeasures, 156
mo_errormeasures::mae, 818
mae_dp_3d
mo_errormeasures, 157
mo_errormeasures::mae, 819
mae_sp_1d
mo_errormeasures, 157
mo_errormeasures::mae, 819
mae_sp_2d
mo_errormeasures, 158
mo_errormeasures::mae, 819
mae_sp_3d
mo_errormeasures, 158
mo_errormeasures::mae, 819
majority_statistics
mo_upscaling_operators, 724
mapcoordinates
mo_grid, 196
mask
mo_common_variables::grid, 798
mo_mrm_write_fluxes_states::outputvariable, 886
mo_write_fluxes_states::outputvariable, 891
max_surfresist
mo_mpr_constants, 273
max_timers
mo_timer, 722
maxgeounit
mo_mpr_constants, 273
maximummonthlyflow
mo_mrm_signatures, 431
maxlen
mo_ncwrite, 513
maxnlcovers
mo_common_constants, 99
maxnobasins
mo_common_constants, 99
maxnogauges
mo_mrm_constants, 324
maxnosoilhorizons
mo_mpr_constants, 273
mcmc_dp
mo_mcmc, 236
mo_mcmc::mcmc, 823
mcmc_error_params

mo_common_mhm_mrm_variables, 110
mcmc_opti
mo_common_mhm_mrm_variables, 110
mcmc_stddev_dp
mo_mcmc, 236
mo_mcmc::mcmc_stddev, 827
mdds
mo_dds, 148
mean_dp
mo_moment, 268
mo_moment::mean, 828
mo_template, 709
mo_template::mean, 829
mean_sp
mo_moment, 268
mo_moment::mean, 828
mo_template, 709
mo_template::mean, 829
median_dp
mo_percentile, 630
mo_percentile::median, 829
median_sp
mo_percentile, 631
mo_percentile::median, 830
mend
mo_common_variables::period, 901
message
mo_message, 238
message_text
mo_message, 240
meteo_forcings_wrapper
mo_meteo_forcings, 244
meteo_weights_wrapper
mo_meteo_forcings, 245
mhm
mo_mhm, 249
mhm_details
mo_common_variables, 128
mhm_driver
mhm_driver.f90, 959
mhm_driver.f90, 959
mhm_driver, 959
mhm_eval
mo_mhm_eval, 259
mhm_initialize
mo_startup, 699
mhm_papers.md, 962
mhm_read_config
mo_mhm_read_config, 262
missing
mo_nml, 593
mixed_central_moment_dp
mo_moment, 268
mo_moment::mixed_central_moment, 830
mixed_central_moment_sp
mo_moment, 268
mo_moment::mixed_central_moment, 830
mixed_central_moment_var_dp
mo_moment, 269
mo_moment::mixed_central_moment_var, 831
mixed_central_moment_var_sp
mo_moment, 269
mo_moment::mixed_central_moment_var, 831
mo_anneal, 83
anneal_dp, 83
dchange_dp, 84
generate_neighborhood_weight_dp, 85
gettemperature_dp, 85
pargen_anneal_dp, 86
pargen_dds_dp, 86
mo_anneal.f90, 962
mo_anneal::anneal, 753
anneal_dp, 755
mo_anneal::generate_neighborhood_weight, 787
generate_neighborhood_weight_dp, 787
mo_anneal::gettemperature, 794
gettemperature_dp, 795
mo_append, 87
append_char_3d, 88
append_char_m_m, 88
append_char_v_s, 89
append_char_v_v, 89
append_dp_3d, 89
append_dp_m_m, 89
append_dp_v_s, 89
append_dp_v_v, 89
append_i4_3d, 90
append_i4_m_m, 90
append_i4_v_s, 90
append_i4_v_v, 90
append_i8_3d, 90
append_i8_m_m, 90
append_i8_v_s, 90
append_i8_v_v, 91
append_lgt_3d, 91
append_lgt_m_m, 91
append_lgt_v_s, 91
append_lgt_v_v, 91
append_sp_3d, 91
append_sp_m_m, 92
append_sp_v_s, 92
append_sp_v_v, 92
paste_char_m_m, 92
paste_char_m_s, 92
paste_char_m_v, 92
paste_dp_m_m, 92
paste_dp_m_s, 93
paste_dp_m_v, 93
paste_i4_m_m, 93
paste_i4_m_s, 93
paste_i4_m_v, 93
paste_i8_m_m, 93
paste_i8_m_s, 93
paste_i8_m_v, 94
paste_lgt_m_m, 94
paste_lgt_m_s, 94

paste_lgt_m_v, 94
paste_sp_m_m, 94
paste_sp_m_s, 94
paste_sp_m_v, 94
mo_append.f90, 963
mo_append::append, 756
append_char_3d, 758
append_char_m_m, 758
append_char_v_s, 758
append_char_v_v, 758
append_dp_3d, 758
append_dp_m_m, 759
append_dp_v_s, 759
append_dp_v_v, 759
append_i4_m_m, 759
append_i4_v_s, 759
append_i4_v_v, 759
append_i8_3d, 760
append_i8_m_m, 760
append_i8_v_s, 760
append_i8_v_v, 760
append_lgt_3d, 760
append_lgt_m_m, 760
append_lgt_v_s, 760
append_lgt_v_v, 761
append_sp_3d, 761
append_sp_m_m, 761
append_sp_v_s, 761
append_sp_v_v, 761
mo_append::paste, 893
paste_char_m_m, 894
paste_char_m_s, 894
paste_char_m_v, 895
paste_dp_m_m, 895
paste_dp_m_s, 895
paste_dp_m_v, 895
paste_i4_m_m, 895
paste_i4_m_s, 895
paste_i4_m_v, 895
paste_i8_m_m, 896
paste_i8_m_s, 896
paste_i8_m_v, 896
paste_lgt_m_m, 896
paste_lgt_m_s, 896
paste_lgt_m_v, 896
paste_sp_m_m, 897
paste_sp_m_s, 897
paste_sp_m_v, 897
mo_canopy_interc, 95
canopy_interc, 95
mo_canopy_interc.f90, 964
mo_common_constants, 97
dayhours, 98
daysecs, 98
eps_dp, 98
eps_sp, 98
hoursecs, 98
maxnlcovers, 99
maxnobasins, 99
ncolpars, 99
nodata_dp, 99
nodata_i4, 100
p1_initstatefluxes, 100
yeardays, 100
yearmonths, 100
yearmonths_i4, 100
mo_common_constants.f90, 964
mo_common_file, 100
file_config, 101
file_dem, 101
uconfig, 101
udem, 101
ulcoverclass, 102
mo_common_file.f90, 965
mo_common_functions, 102
in_bound, 102
mo_common_functions.f90, 965
mo_common_mHM_mRM_file.f90, 966
mo_common_mHM_mRM_read_config.f90, 966
mo_common_mHM_mRM_variables.f90, 966
mo_common_mhm_mrm_file, 103
file_opti, 103
file_opti_nml, 104
uopti, 104
uopti_nml, 104
mo_common_mhm_mrm_read_config, 104
check_optimization_settings, 105
common_check_resolution, 106
common_mhm_mrm_read_config, 106
mo_common_mhm_mrm_variables, 108
c2tstu, 109
dds_r, 109
dirrestartin, 110
evalper, 110
lcyearid, 110
mcmc_error_params, 110
mcmc_opti, 110
mrm_coupling_mode, 110
nerror_model, 111
niterations, 111
ntstepday, 111
opti_function, 111
opti_method, 111
optimize, 111
optimize_restart, 112
read_restart, 112
readper, 112
resolutionrouting, 112
sa_temp, 112
sce_ngs, 112
sce_npg, 113
sce_nps, 113
seed, 113
simper, 113
timestep, 113
warmingdays, 113

warmper, 114
mo_common_read_config, 114
common_read_config, 114
set_land_cover_scenes_id, 116
mo_common_read_config.f90, 967
mo_common_read_data, 117
read_dem, 117
read_lcover, 118
mo_common_read_data.f90, 967
mo_common_restart, 119
read_grid_info, 119
write_grid_info, 121
mo_common_restart.f90, 968
mo_common_variables, 122
alma_convention, 123
contact, 123
conventions, 123
dircommonfiles, 124
dirconfigout, 124
dirlcover, 124
dirmorpho, 124
dirout, 124
dirrestartout, 124
filelatlon, 125
global_parameters, 125
global_parameters_name, 125
history, 125
iflag_cordinate_sys, 125
l0_basin, 125
l0_elev, 126
l0_l11_remap, 126
l0_l1_remap, 126
l0_lcover, 126
l11_basin, 126
l1_l11_remap, 126
lc_year_end, 126
lc_year_start, 127
lcfilename, 127
level0, 127
level1, 127
mhm_details, 128
nbasins, 128
nlcoverscene, 128
nprocesses, 128
nuniquel0basins, 128
processmatrix, 129
project_details, 129
resolutionhydrology, 129
setup_description, 129
simulation_type, 129
write_restart, 129
mo_common_variables.f90, 968
mo_common_variables::grid, 797
cellarea, 797
cellcoor, 797
cellsize, 798
id, 798
iend, 798
istart, 798
mask, 798
ncells, 798
ncols, 798
nodata_value, 798
nrows, 798
x, 799
xllcorner, 799
y, 799
yllcorner, 799
mo_common_variables::gridremapper, 800
high_res_grid, 800
left_bound, 801
low_res_grid, 801
lower_bound, 801
lowres_id_on_highres, 801
n_subcells, 801
right_bound, 801
upper_bound, 801
mo_common_variables::period, 900
dend, 901
dstart, 901
julend, 901
julstart, 901
mend, 901
mstart, 901
nobs, 901
yend, 901
ystart, 902
mo_constants, 130
cp0_dp, 132
cp0_sp, 132
dayhours, 132
daysecs, 132
deg2rad_dp, 132
deg2rad_sp, 133
euler, 133
euler_d, 133
gravity_dp, 133
gravity_sp, 133
nerr, 133
nin, 133
nnml, 134
nout, 134
p0_dp, 134
p0_sp, 134
pi, 134
pi_d, 134
pi_dp, 134
pi_sp, 135
pio2, 135
pio2_d, 135
pio2_dp, 135
pio2_sp, 135
psychro_dp, 135
psychro_sp, 135
rad2deg_dp, 136
rad2deg_sp, 136

radiusearth_dp, 136
radiusearth_sp, 136
rho0_dp, 136
rho0_sp, 136
secday_dp, 136
secday_sp, 136
sigma_dp, 137
sigma_sp, 137
solarconst_dp, 137
solarconst_sp, 137
specheatet_dp, 137
specheatet_sp, 137
sqrt2, 137
sqrt2_d, 138
sqrt2_dp, 138
sqrt2_sp, 138
t0_dp, 138
t0_sp, 138
twopi, 138
twopi_d, 138
twopi_dp, 138
twopi_sp, 139
twothird_dp, 139
twothird_sp, 139
yeardays, 139
yearmonths, 139
mo_constants.f90, 969
mo_corr, 139
arth_dp, 140
arth_i4, 140
arth_sp, 141
autocoeffk_1d_dp, 141
autocoeffk_1d_sp, 141
autocoeffk_dp, 141
autocoeffk_sp, 141
autocorr_1d_dp, 141
autocorr_1d_sp, 142
autocorr_dp, 142
autocorr_sp, 142
corr_dp, 142
corr_sp, 142
crosscoeffk_dp, 142
crosscoeffk_sp, 143
crosscorr_dp, 143
crosscorr_sp, 143
four1_dp, 143
four1_sp, 143
fourrow_dp, 143
fourrow_sp, 144
npar2_arth, 146
npar_arth, 146
realft_dp, 144
realft_sp, 144
swap_1d_dpc, 145
swap_1d_spc, 145
zroots_unity_dp, 145
zroots_unity_sp, 145
mo_corr.f90, 971
mo_corr::arth, 761
arth_dp, 762
arth_i4, 762
arth_sp, 762
mo_corr::autocoeffk, 764
autocoeffk_1d_dp, 764
autocoeffk_1d_sp, 764
autocoeffk_dp, 764
autocoeffk_sp, 764
mo_corr::autocorr, 765
autocorr_1d_dp, 765
autocorr_1d_sp, 765
autocorr_dp, 765
autocorr_sp, 765
mo_corr::corr, 772
corr_dp, 772
corr_sp, 773
mo_corr::crosscoeffk, 774
crosscoeffk_dp, 774
crosscoeffk_sp, 774
mo_corr::crosscorr, 775
crosscorr_dp, 775
crosscorr_sp, 775
mo_corr::four1, 785
four1_dp, 785
four1_sp, 785
mo_corr::fourrow, 785
fourrow_dp, 785
fourrow_sp, 785
mo_corr::realft, 905
realft_dp, 905
realft_sp, 905
mo_corr::swap, 930
swap_1d_dpc, 930
swap_1d_spc, 930
mo_dds, 146
dds, 147
mdds, 148
neigh_value, 150
mo_dds.f90, 972
mo_errormeasures, 150
bias_dp_1d, 152
bias_dp_2d, 152
bias_dp_3d, 152
bias_sp_1d, 152
bias_sp_2d, 153
bias_sp_3d, 153
kge_dp_1d, 153
kge_dp_2d, 153
kge_dp_3d, 153
kge_sp_1d, 153
kge_sp_2d, 154
kge_sp_3d, 154
kgenocorr_dp_1d, 154
kgenocorr_dp_2d, 154
kgenocorr_dp_3d, 154
kgenocorr_sp_1d, 154
kgenocorr_sp_2d, 155

kgenocorr_sp_3d, 155
lnnse_dp_1d, 155
lnnse_dp_2d, 155
lnnse_dp_3d, 155
lnnse_sp_1d, 155
lnnse_sp_2d, 155
lnnse_sp_3d, 156
mae_dp_1d, 156
mae_dp_2d, 156
mae_dp_3d, 157
mae_sp_1d, 157
mae_sp_2d, 158
mae_sp_3d, 158
mse_dp_1d, 158
mse_dp_2d, 159
mse_dp_3d, 160
mse_sp_1d, 160
mse_sp_2d, 161
mse_sp_3d, 162
nse_dp_1d, 162
nse_dp_2d, 163
nse_dp_3d, 163
nse_sp_1d, 163
nse_sp_2d, 163
nse_sp_3d, 163
rmse_dp_1d, 163
rmse_dp_2d, 164
rmse_dp_3d, 164
rmse_sp_1d, 164
rmse_sp_2d, 165
rmse_sp_3d, 165
sae_dp_1d, 166
sae_dp_2d, 166
sae_dp_3d, 167
sae_sp_1d, 168
sae_sp_2d, 168
sae_sp_3d, 169
sse_dp_1d, 170
sse_dp_2d, 170
sse_dp_3d, 171
sse_sp_1d, 172
sse_sp_2d, 172
sse_sp_3d, 173
wnse_dp_1d, 174
wnse_dp_2d, 174
wnse_dp_3d, 174
wnse_sp_1d, 174
wnse_sp_2d, 174
wnse_sp_3d, 174
mo_errormeasures.f90, 973
mo_errormeasures::bias, 768
bias_dp_1d, 768
bias_dp_2d, 769
bias_dp_3d, 769
bias_sp_1d, 769
bias_sp_2d, 769
bias_sp_3d, 769
mo_errormeasures::kge, 807
kge_dp_1d, 808
kge_dp_2d, 808
kge_dp_3d, 809
kge_sp_1d, 809
kge_sp_2d, 809
kge_sp_3d, 809
mo_errormeasures::kgenocorr, 809
kgenocorr_dp_1d, 810
kgenocorr_dp_2d, 810
kgenocorr_dp_3d, 811
kgenocorr_sp_1d, 811
kgenocorr_sp_2d, 811
kgenocorr_sp_3d, 811
mo_errormeasures::lnnse, 814
lnnse_dp_1d, 815
lnnse_dp_2d, 815
lnnse_dp_3d, 815
lnnse_sp_1d, 815
lnnse_sp_2d, 815
lnnse_sp_3d, 815
mo_errormeasures::mae, 818
mae_dp_1d, 818
mae_dp_2d, 818
mae_dp_3d, 819
mae_sp_1d, 819
mae_sp_2d, 819
mae_sp_3d, 819
mo_errormeasures::mse, 833
mse_dp_1d, 834
mse_dp_2d, 834
mse_dp_3d, 834
mse_sp_1d, 834
mse_sp_2d, 834
mse_sp_3d, 834
mo_errormeasures::nse, 871
nse_dp_1d, 871
nse_dp_2d, 871
nse_dp_3d, 871
nse_sp_1d, 871
nse_sp_2d, 872
nse_sp_3d, 872
mo_errormeasures::rmse, 908
rmse_dp_1d, 908
rmse_dp_2d, 908
rmse_dp_3d, 909
rmse_sp_1d, 909
rmse_sp_2d, 909
rmse_sp_3d, 909
mo_errormeasures::sae, 910
sae_dp_1d, 910
sae_dp_2d, 910
sae_dp_3d, 911
sae_sp_1d, 911
sae_sp_2d, 911
sae_sp_3d, 911
mo_errormeasures::sse, 926
sse_dp_1d, 927
sse_dp_2d, 927

sse_dp_3d, 927
sse_sp_1d, 927
sse_sp_2d, 927
sse_sp_3d, 927
mo_errormeasures::wnse, 954
wnse_dp_1d, 954
wnse_dp_2d, 954
wnse_dp_3d, 954
wnse_sp_1d, 954
wnse_sp_2d, 954
wnse_sp_3d, 955
mo_file, 175
file_defoutput, 175
file_main, 176
file_namelist_mhm, 176
file_namelist_mhm_param, 176
udefoutput, 176
unamelist_mhm, 176
unamelist_mhm_param, 176
utws, 176
version, 177
version_date, 177
mo_file.f90, 974
mo_finish, 177
finish, 177
mo_finish.f90, 975
mo_global_variables, 179
basin_avg_tws_obs, 180
basin_avg_tws_sim, 181
dirabsvappressure, 181
direvapotranspiration, 181
dirmaxtemperature, 181
dirmintemperature, 181
dirnetradiation, 181
dirneutrons, 181
dirprecipitation, 182
dirreferenceet, 182
dirsoil_moisture, 182
dirtemperature, 182
dirwindspeed, 182
evap_coeff, 182
fday_pet, 182
fday_prec, 183
fday_temp, 183
filetws, 183
fnight_pet, 183
fnight_prec, 183
fnight_temp, 183
inputformat_meteo_forcings, 183
l1_absvappress, 184
l1_aetcanopy, 184
l1_aetsealed, 184
l1_aetsoil, 184
l1_baseflow, 184
l1_et, 184
l1_et_mask, 184
l1_fastrunoff, 185
l1_infilsoil, 185
l1_inter, 185
l1_melt, 185
l1_netrad, 185
l1_neutrons, 185
l1_neutronsdata, 185
l1_neutronsdata_mask, 186
l1_percol, 186
l1_pet, 186
l1_pet_calc, 186
l1_pet_weights, 186
l1_pre, 186
l1_pre_weights, 186
l1_preeffect, 187
l1_rain, 187
l1_runoffseal, 187
l1_satstw, 187
l1_sealstw, 187
l1_slowrunoff, 187
l1_sm, 187
l1_sm_mask, 188
l1_snow, 188
l1_snowpack, 188
l1_soilmoist, 188
l1_temp, 188
l1_temp_weights, 188
l1_throughfall, 189
l1_tmax, 189
l1_tmin, 189
l1_total_runoff, 189
l1_unsatstw, 189
l1_windspeed, 189
level2, 189
neutron_integral_afast, 190
nmeasperday_tws, 190
nsoilhorizons_sm_input, 190
ntimesteps_l1_et, 190
ntimesteps_l1_neutrons, 190
ntimesteps_l1_sm, 190
outputflxstate, 190
read_meteo_weights, 190
routingstates, 191
timestep_et_input, 191
timestep_model_inputs, 191
timestep_model_outputs, 191
timestep_neutrons_input, 191
timestep_sm_input, 191
mo_global_variables.f90, 975
mo_global_variables::twsstructure, 932
basinid, 932
fname, 932
tws, 933
mo_grid, 192
calculate_grid_properties, 192
geocoordinates, 194
init_lowres_level, 194
l0_grid_setup, 195
mapcoordinates, 196
set_basin_indices, 197

mo_grid.f90, 977
mo_init_states, 198
variables_alloc, 198
variables_default_init, 200
mo_init_states.f90, 977
mo_julian, 201
caldat, 203
caldat360, 205
caldat365, 205
caldatjulian, 206
calendar, 229
date2dec, 208
date2dec360, 209
date2dec365, 210
date2decjulian, 211
dec2date, 212
dec2date360, 214
dec2date365, 216
dec2datejulian, 217
julday, 219
julday360, 220
julday365, 221
juldayjulian, 223
ndays, 224
ndyin, 225
selectcalendar, 226
setcalendarinteger, 227
setcalendarstring, 228
mo_julian.f90, 978
mo_julian::setcalendar, 911
setcalendarinteger, 912
setcalendarstring, 912
mo_kind, 229
dp, 230
dpc, 230
i1, 230
i2, 230
i4, 231
i8, 231
lgt, 231
sp, 231
spc, 232
mo_kind.f90, 979
mo_kind::sprs2_dp, 924
irow, 924
jcol, 924
len, 925
n, 925
val, 925
mo_kind::sprs2_sp, 925
irow, 926
jcol, 926
len, 926
n, 926
val, 926
mo_linfit, 232
linfit_dp, 232
linfit_sp, 233
mo_linfit.f90, 980
mo_linfit::linfit, 813
linfit_dp, 814
linfit_sp, 814
mo_mad, 233
mad_dp, 233
mad_sp, 233
mad_val_dp, 234
mad_val_sp, 234
mo_mad.f90, 980
mo_mad::mad, 817
mad_dp, 817
mad_sp, 817
mad_val_dp, 818
mad_val_sp, 818
mo_mcmc, 234
generatenewparameterset_dp, 235
mcmc_dp, 236
mcmc_stddev_dp, 236
pargen_dp, 237
pargennorm_dp, 237
mo_mcmc.f90, 980
mo_mcmc::mcmc, 819
mcmc_dp, 823
mo_mcmc::mcmc_stddev, 824
mcmc_stddev_dp, 827
mo_message, 237
message, 238
message_text, 240
mo_message.f90, 981
mo_meteo_forcings, 240
chunk_config, 240
chunk_size, 242
is_read, 243
meteo_forcings_wrapper, 244
meteo_weights_wrapper, 245
prepare_meteo_forcings_data, 247
mo_meteo_forcings.f90, 981
mo_mhm, 248
mhm, 249
mo_mhm.f90, 982
mo_mhm_constants, 254
c1_initstatesm, 255
cosmic_alpha, 255
cosmic_bd, 255
cosmic_l1, 255
cosmic_l2, 255
cosmic_l3, 255
cosmic_l4, 256
cosmic_n, 256
cosmic_vwclat, 256
desilets_a0, 256
desilets_a1, 256
desilets_a2, 256
duffiedelta1, 256
duffiedelta2, 257
duffiedr, 257
h2odens, 257

harsamconst, 257
noutflxstate, 257
p2_initstatefluxes, 257
p3_initstatefluxes, 257
p4_initstatefluxes, 257
p5_initstatefluxes, 258
satpressureslope1, 258
stboltzmann, 258
tetens_c1, 258
tetens_c2, 258
tetens_c3, 258
mo_mhm_constants.f90, 982
mo_mhm_eval, 258
mhm_eval, 259
mo_mhm_eval.f90, 983
mo_mhm_read_config, 262
mhm_read_config, 262
mo_mhm_read_config.f90, 983
mo_moment, 265
absdev_dp, 266
absdev_sp, 266
average_dp, 266
average_sp, 266
central_moment_dp, 266
central_moment_sp, 266
central_moment_var_dp, 267
central_moment_var_sp, 267
correlation_dp, 267
correlation_sp, 267
covariance_dp, 267
covariance_sp, 267
kurtosis_dp, 268
kurtosis_sp, 268
mean_dp, 268
mean_sp, 268
mixed_central_moment_dp, 268
mixed_central_moment_sp, 268
mixed_central_moment_var_dp, 269
mixed_central_moment_var_sp, 269
moment_dp, 269
moment_sp, 269
skewness_dp, 269
skewness_sp, 270
stddev_dp, 270
stddev_sp, 270
variance_dp, 270
variance_sp, 270
mo_moment.f90, 983
mo_moment::absdev, 753
absdev_dp, 753
absdev_sp, 753
mo_moment::average, 765
average_dp, 766
average_sp, 766
mo_moment::central_moment, 770
central_moment_dp, 770
central_moment_sp, 770
mo_moment::central_moment_var, 770
central_moment_var_dp, 770
central_moment_var_sp, 770
mo_moment::correlation, 773
correlation_dp, 773
correlation_sp, 773
mo_moment::covariance, 773
covariance_dp, 774
covariance_sp, 774
mo_moment::kurtosis, 811
kurtosis_dp, 812
kurtosis_sp, 812
mo_moment::mean, 828
mean_dp, 828
mean_sp, 828
mo_moment::mixed_central_moment, 830
mixed_central_moment_dp, 830
mixed_central_moment_sp, 830
mo_moment::mixed_central_moment_var, 831
mixed_central_moment_var_dp, 831
mixed_central_moment_var_sp, 831
mo_moment::moment, 831
moment_dp, 831
moment_sp, 832
mo_moment::skewness, 912
skewness_dp, 913
skewness_sp, 913
mo_moment::stddev, 929
stddev_dp, 929
stddev_sp, 929
mo_moment::variance, 953
variance_dp, 953
variance_sp, 953
mo_mpr_constants, 270
bulkdens_orgmatter, 272
c1_initstatesm, 272
field_cap_c1, 272
field_cap_c2, 272
karman, 272
ks_c, 272
lai_factor_surfresi, 272
lai_offset_surfresi, 273
max_surfresist, 273
maxgeounit, 273
maxnosoilhorizons, 273
nlcover_class, 273
p2_initstatefluxes, 273
p3_initstatefluxes, 273
p4_initstatefluxes, 273
p5_initstatefluxes, 274
pwp_c, 274
pwp_matpot_thetar, 274
vgenuchten_sandtresh, 274
vgenuchtenn_c1, 274
vgenuchtenn_c10, 274
vgenuchtenn_c11, 274
vgenuchtenn_c12, 275
vgenuchtenn_c13, 275
vgenuchtenn_c14, 275

vgenuchtenn_c15, 275
vgenuchtenn_c16, 275
vgenuchtenn_c17, 275
vgenuchtenn_c18, 275
vgenuchtenn_c2, 275
vgenuchtenn_c3, 276
vgenuchtenn_c4, 276
vgenuchtenn_c5, 276
vgenuchtenn_c6, 276
vgenuchtenn_c7, 276
vgenuchtenn_c8, 276
vgenuchtenn_c9, 276
windmeasheight, 277
mo_mpr_constants.f90, 985
mo_mpr_eval, 277
mpr_eval, 277
mo_mpr_eval.f90, 986
mo_mpr_file, 280
file_aspect, 281
file_geolut, 281
file_hydrogeoclass, 281
file_laiclass, 281
file_lailut, 282
file_main, 282
file_meteo_binary_end, 282
file_meteo_header, 282
file_namelist_mpr, 282
file_namelist_mpr_param, 282
file_slope, 282
file_soil_database, 283
file_soil_database_1, 283
file_soilclass, 283
uaspect, 283
ugeolut, 283
uhydrogeoclass, 283
ulaiclass, 284
ulailut, 284
umeteo, 284
umeteo_header, 284
unamelist_mpr, 284
unamelist_mpr_param, 284
uslope, 285
usoil_database, 285
usoilclass, 285
version, 285
version_date, 285
mo_mpr_file.f90, 986
mo_mpr_global_variables, 285
dirgridded_lai, 287
fracsealed_cityarea, 287
geounitkar, 287
geounitlist, 287
horizondepth_mhm, 287
iflag_soildb, 287
inputformat_gridded_lai, 288
l0_asp, 288
l0_geounit, 288
l0_gridded_lai, 288
l0_slope, 288
l0_slope_emp, 288
l0_soilid, 289
l1_aeroresist, 289
l1_alpha, 289
l1_degday, 289
l1_degdayinc, 289
l1_degdaymax, 289
l1_degdaynopre, 289
l1_fasp, 290
l1_froots, 290
l1_fsealed, 290
l1_harsamcoeff, 290
l1_jarvis_thresh_c1, 290
l1_karstloss, 290
l1_kbaseflow, 290
l1_kfastflow, 291
l1_kperco, 291
l1_kslowflow, 291
l1_maxinter, 291
l1_petlaicorfactor, 291
l1_prietayalpha, 291
l1_sealedthresh, 292
l1_soilmoistexp, 292
l1_soilmoistfc, 292
l1_soilmoistsat, 292
l1_surfresist, 292
l1_tempthresh, 292
l1_unsatthresh, 292
l1_wiltingpoint, 293
lailut, 293
laiper, 293
laiunitlist, 293
ngeounits, 293
nlai, 293
nlaiclass, 293
nsoilhorizons_mhm, 294
nsoiltypes, 294
soildb, 294
tillagedepth, 294
timestep_lai_input, 294
mo_mpr_global_variables.f90, 987
mo_mpr_global_variables::soiltype, 913
clay, 914
db, 914
dbm, 914
depth, 914
id, 914
is_present, 914
ks, 915
ld, 915
nhorizons, 915
ntillhorizons, 915
rzdepth, 915
sand, 915
thetafc, 915
thetafc_till, 915
thetapw, 915

thetapw_till, 916
thetas, 916
thetas_till, 916
ud, 916
wd, 916
mo_mpr_pet, 294
bulksurface_resistance, 295
pet_correctbyasp, 296
pet_correctbylai, 297
priestley_taylor_alpha, 299
mo_mpr_pet.f90, 988
mo_mpr_read_config, 300
mpr_read_config, 300
mo_mpr_read_config.f90, 989
mo_mpr_restart, 302
unpack_field_and_write_1d_dp, 303
unpack_field_and_write_1d_i4, 303
unpack_field_and_write_2d_dp, 303
unpack_field_and_write_3d_dp, 304
write_eff_params, 304
write_mpr_restart_files, 305
mo_mpr_restart.f90, 989
mo_mpr_restart::unpack_field_and_write, 937
unpack_field_and_write_1d_dp, 938
unpack_field_and_write_1d_i4, 938
unpack_field_and_write_2d_dp, 939
unpack_field_and_write_3d_dp, 939
mo_mpr_runoff, 306
mpr_runoff, 307
mo_mpr_runoff.f90, 990
mo_mpr_smhorizons, 308
mpr_smhorizons, 309
mo_mpr_smhorizons.f90, 990
mo_mpr_soilmoist, 311
field_cap, 312
genuchten, 313
hydro_cond, 314
mpr_sm, 315
pwp, 317
mo_mpr_soilmoist.f90, 990
mo_mpr_startup, 318
init_eff_params, 319
l0_check_input, 319
l0_variable_init, 320
mpr_initialize, 322
mo_mpr_startup.f90, 991
mo_mrm_constants, 323
deltah, 324
given_ts, 324
maxnogauges, 324
noutflxstate, 324
nroutingstates, 324
rout_space_weight, 325
mo_mrm_constants.f90, 991
mo_mrm_eval, 325
mrm_eval, 325
mo_mrm_eval.f90, 991
mo_mrm_file, 327
file_config, 329
file_daily_discharge, 329
file_defoutput, 329
file_facc, 329
file_fdir, 329
file_gaugeloc, 330
file_gw_output, 330
file_main, 330
file_mrm_output, 330
file_namelist_mrm, 330
file_namelist_param_mrm, 330
file_slope, 330
ncfile_discharge, 331
uconfig, 331
udaily_discharge, 331
udefoutput, 331
udischarge, 331
ufacc, 331
ufdir, 332
ugaugeloc, 332
unamelist_mrm, 332
unamelist_param_mrm, 332
uslope, 332
version, 332
version_date, 332
mo_mrm_file.f90, 992
mo_mrm_global_variables, 333
basin_mrm, 334
dirbankfullrunoff, 335
dirgauges, 335
dirtotalrunoff, 335
filenametotalrunoff, 335
gauge, 335
gw_coupling, 335
inflowgauge, 336
is_start, 336
l0_celerity, 336
l0_channel_depth, 336
l0_channel_elevation, 336
l0_dracell, 336
l0_drasc, 336
l0_facc, 337
l0_fdir, 337
l0_floodplain, 337
l0_gaugeloc, 337
l0_inflowgaugeloc, 337
l0_l11_remap, 337
l0_noutlet, 338
l0_river_head_mon_sum, 338
l0_slope, 338
l0_streamnet, 338
l11_afloodplain, 338
l11_areacell, 338
l11_bankfull_runoff_in, 338
l11_c1, 339
l11_c2, 339
l11_celerity, 339
l11_cellcoor, 339

l11_colout, 339
l11_facc, 339
l11_fcol, 339
l11_fdir, 340
l11_fromn, 340
l11_frow, 340
l11_k, 340
l11_l1_id, 340
l11_label, 340
l11_length, 341
l11_linkin_facc, 341
l11_meandering, 341
l11_netperm, 341
l11_nlinkfracfpimp, 341
l11_noutlets, 341
l11_qmod, 341
l11_qout, 342
l11_qtin, 342
l11_qtr, 342
l11_rorder, 342
l11_rowout, 342
l11_sink, 342
l11_slope, 343
l11_tcol, 343
l11_ton, 343
l11_trow, 343
l11_tsrout, 343
l11_xi, 343
l1_l11_id, 343
l1_l11_remap, 344
l1_total_runoff_in, 344
level11, 344
mrm_runoff, 344
ngaugestotal, 344
ninflowgaugestotal, 344
nmeasperday, 345
ntstepday, 345
outputflxstate_mrm, 345
timestep_model_outputs_mrm, 345
varnametotalrunoff, 345
mo_mrm_global_variables.f90, 993
mo_mrm_global_variables::basininfo_mrm, 766
gaugeidlist, 767
gaugeindexlist, 767
gaugenodelist, 767
inflowgaugeheadwater, 767
inflowgaugeidlist, 767
inflowgaugeindexlist, 767
inflowgaugenodelist, 767
l0_coloutlet, 768
l0_noutlet, 768
l0_rowoutlet, 768
ngauges, 768
ninflowgauges, 768
mo_mrm_global_variables::gaugingstation, 786
basinid, 786
fname, 786
gaugeid, 786
q, 787
mo_mrm_init, 345
config_output, 346
l0_check_input_routing, 347
mrm_init, 348
print_startup_message, 351
variables_alloc_routing, 352
variables_default_init_routing, 353
mo_mrm_init.f90, 995
mo_mrm_mpr, 353
mrm_init_param, 354
mrm_update_param, 355
reg_rout, 356
mo_mrm_mpr.f90, 995
mo_mrm_net_startup, 358
celllength, 359
get_distance_two_lat_lon_points, 360
l11_calc_celerity, 361
l11_flow_accumulation, 362
l11_flow_direction, 363
l11_fraction_sealed_floodplain, 365
l11_l1_mapping, 366
l11_link_location, 367
l11_routing_order, 368
l11_set_drain_outlet_gauges, 369
l11_set_network_topology, 370
l11_stream_features, 371
movedownonecell, 372
moveup, 373
mo_mrm_net_startup.f90, 995
calculate_l11_flow_accumulation, 996
mo_mrm_objective_function_runoff, 374
extract_runoff, 375
loglikelihood_evin2013_2, 377
loglikelihood_stddev, 378
loglikelihood_trend_no_autocorr, 380
multi_objective_ae_fdc_lsv_nse_djf, 381
multi_objective_lnnse_highflow_lnnse_lowflow, 382
multi_objective_lnnse_highflow_lnnse_lowflow_2, 384
multi_objective_nse_lnnse, 385
multi_objective_runoff, 386
objective_equal_nse_lnnse, 387
objective_kge, 389
objective_lnnse, 390
objective_multiple_gauges_kge_power6, 392
objective_nse, 393
objective_power6_nse_lnnse, 394
objective_sse, 395
objective_weighted_nse, 397
parameter_regularization, 398
single_objective_runoff, 399
mo_mrm_objective_function_runoff.f90, 997
mo_mrm_read_config, 400
mrm_read_config, 401
read_mrm_routing_params, 402
mo_mrm_read_config.f90, 998

mo_mrm_read_data, 404
mrm_read_bankfull_runoff, 404
mrm_read_discharge, 405
mrm_read_l0_data, 406
mrm_read_total_runoff, 407
rotate_fdir_variable, 408
mo_mrm_read_data.f90, 998
mo_mrm_restart, 409
mrm_read_restart_config, 410
mrm_read_restart_states, 411
mrm_write_restart, 412
mo_mrm_restart.f90, 999
mo_mrm_river_head, 413
avg_and_write_timestep, 414
calc_channel_elevation, 414
calc_river_head, 415
calc_slope, 416
create_output, 416
init_masked_zeros_l0, 417
nc_riverhead, 418
nc_time, 419
reset_sum, 418
sum_counter, 419
time_counter, 419
mo_mrm_river_head.f90, 999
mo_mrm_routing, 419
add_inflow, 420
l11_routing, 421
l11_runoff_acc, 423
mrm_routing, 424
mo_mrm_routing.f90, 1000
mo_mrm_signatures, 427
autocorrelation, 428
flowdurationcurve, 428
limb_densities, 430
maximummonthlyflow, 431
moments, 432
peakdistribution, 433
runoffratio, 434
zeroflowratio, 435
mo_mrm_signatures.f90, 1000
mo_mrm_write, 436
average_counter, 445
day_counter, 446
month_counter, 446
mrm_write, 437
mrm_write_optifile, 438
mrm_write_optinamelist, 439
mrm_write_output_fluxes, 441
nc, 446
write_configfile, 443
write_daily_obs_sim_discharge, 444
year_counter, 446
mo_mrm_write.f90, 1001
mo_mrm_write_fluxes_states, 446
close, 447
createoutputfile, 448
newoutputdataset, 449
newoutputvariable, 450
updatedataset, 451
updatevariable, 452
writetimestep, 453
writevariableattributes, 453
writevariabletimestep, 454
mo_mrm_write_fluxes_states.f90, 1001
mo_mrm_write_fluxes_states::outputdataset, 876
all, 877
basin, 877
close, 877
count, 877
counter, 878
created, 878
ibasin, 878
id, 878
nc, 878
ncdataset, 878
steps, 878
store, 878
time, 878
to, 879
updatedataset, 877
variables, 879
vars, 879
write, 879
writetimestep, 877
written, 879
mo_mrm_write_fluxes_states::outputvariable, 884
average, 885
avg, 885
before, 885
between, 886
calls, 886
contains, 886
count, 886
counter, 886
data, 886
mask, 886
nc, 886
ncdataset, 887
number, 887
of, 887
reconstruct, 887
store, 887
the, 887
to, 887
updatevariable, 885, 888
variable, 888
which, 888
writes, 888
writevariabletimestep, 885
writing, 888
mo_multi_param_reg, 455
aerodynamical_resistance, 455
baseflow_param, 457
canopy_intercept_param, 458
iper_thres_runoff, 459

karstic_layer, 460
mpr, 461
snow_acc_melt_param, 466
mo_multi_param_reg.f90, 1002
mo_ncread, 467
check, 468
get_info, 469
get_ncdim, 471
get_ncdimatt, 473
get_ncvar_0d_dp, 474
get_ncvar_0d_i1, 474
get_ncvar_0d_i4, 475
get_ncvar_0d_sp, 475
get_ncvar_1d_dp, 476
get_ncvar_1d_i1, 476
get_ncvar_1d_i4, 477
get_ncvar_1d_sp, 477
get_ncvar_2d_dp, 478
get_ncvar_2d_i1, 478
get_ncvar_2d_i4, 479
get_ncvar_2d_sp, 479
get_ncvar_3d_dp, 480
get_ncvar_3d_i1, 480
get_ncvar_3d_i4, 481
get_ncvar_3d_sp, 481
get_ncvar_4d_dp, 482
get_ncvar_4d_i1, 482
get_ncvar_4d_i4, 483
get_ncvar_4d_sp, 483
get_ncvar_5d_dp, 484
get_ncvar_5d_i1, 484
get_ncvar_5d_i4, 485
get_ncvar_5d_sp, 485
get_ncvaratt, 486
ncclose, 486
ncopen, 487
mo_ncread.f90, 1003
mo_ncread::get_ncvar, 788
get_ncvar_0d_dp, 788
get_ncvar_0d_i1, 788
get_ncvar_0d_i4, 789
get_ncvar_0d_sp, 789
get_ncvar_1d_dp, 789
get_ncvar_1d_i1, 789
get_ncvar_1d_i4, 789
get_ncvar_1d_sp, 790
get_ncvar_2d_dp, 790
get_ncvar_2d_i1, 790
get_ncvar_2d_i4, 790
get_ncvar_2d_sp, 790
get_ncvar_3d_dp, 791
get_ncvar_3d_i1, 791
get_ncvar_3d_i4, 791
get_ncvar_3d_sp, 791
get_ncvar_4d_dp, 792
get_ncvar_4d_i1, 792
get_ncvar_4d_i4, 792
get_ncvar_4d_sp, 792
get_ncvar_5d_dp, 792
get_ncvar_5d_i1, 793
get_ncvar_5d_i4, 793
get_ncvar_5d_sp, 793
mo_ncwrite, 487
check, 489
close_netcdf, 490
create_netcdf, 492
dnc, 513
dump_netcdf_1d_dp, 493
dump_netcdf_1d_i4, 493
dump_netcdf_1d_sp, 494
dump_netcdf_2d_dp, 494
dump_netcdf_2d_i4, 495
dump_netcdf_2d_sp, 495
dump_netcdf_3d_dp, 496
dump_netcdf_3d_i4, 496
dump_netcdf_3d_sp, 497
dump_netcdf_4d_dp, 497
dump_netcdf_4d_i4, 498
dump_netcdf_4d_sp, 498
dump_netcdf_5d_dp, 499
dump_netcdf_5d_i4, 499
dump_netcdf_5d_sp, 500
gatt, 513
maxlen, 513
nattdim, 514
ndims, 514
ngatt, 514
nmaxatt, 514
nmaxdim, 514
nvars, 514
open_netcdf, 500
v, 514
var2nc_1d_dp, 502
var2nc_1d_i4, 503
var2nc_1d_sp, 504
var2nc_2d_dp, 504
var2nc_2d_i4, 505
var2nc_2d_sp, 506
var2nc_3d_dp, 506
var2nc_3d_i4, 507
var2nc_3d_sp, 508
var2nc_4d_dp, 508
var2nc_4d_i4, 509
var2nc_4d_sp, 510
var2nc_5d_dp, 510
var2nc_5d_i4, 511
var2nc_5d_sp, 512
write_dynamic_netcdf, 512
write_static_netcdf, 513
mo_ncwrite.f90, 1004
mo_ncwrite::attribute, 763
name, 763
nvalues, 763
values, 763
xtype, 763
mo_ncwrite::dims, 777

dimid, 778
len, 778
name, 778
mo_ncwrite::dump_netcdf, 778
dump_netcdf_1d_dp, 778
dump_netcdf_1d_i4, 779
dump_netcdf_1d_sp, 779
dump_netcdf_2d_dp, 779
dump_netcdf_2d_i4, 779
dump_netcdf_2d_sp, 780
dump_netcdf_3d_dp, 780
dump_netcdf_3d_i4, 780
dump_netcdf_3d_sp, 780
dump_netcdf_4d_dp, 780
dump_netcdf_4d_i4, 781
dump_netcdf_4d_sp, 781
dump_netcdf_5d_dp, 781
dump_netcdf_5d_i4, 781
dump_netcdf_5d_sp, 781
mo_ncwrite::var2nc, 941
var2nc_1d_dp, 942
var2nc_1d_i4, 943
var2nc_1d_sp, 943
var2nc_2d_dp, 943
var2nc_2d_i4, 943
var2nc_2d_sp, 944
var2nc_3d_dp, 944
var2nc_3d_i4, 944
var2nc_3d_sp, 945
var2nc_4d_dp, 945
var2nc_4d_i4, 945
var2nc_4d_sp, 946
var2nc_5d_dp, 946
var2nc_5d_i4, 946
var2nc_5d_sp, 947
mo_ncwrite::variable, 948
att, 949
count, 949
dimids, 949
dimtypes, 949
g0_b, 950
g0_d, 950
g0_f, 950
g0_i, 950
g1_b, 950
g1_d, 950
g1_f, 950
g1_i, 950
g2_b, 950
g2_d, 951
g2_f, 951
g2_i, 951
g3_b, 951
g3_d, 951
g3_f, 951
g3_i, 951
g4_b, 951
g4_d, 951
g4_f, 952
g4_i, 952
name, 952
natt, 952
ndims, 952
nlvls, 952
nsubs, 952
start, 952
unlimited, 952
varid, 953
wflag, 953
xtype, 953
mo_netcdf, 515
check, 518
close, 518
equalncdimensions, 519
getdata1df32, 519
getdata1df64, 519
getdata1di16, 520
getdata1di32, 520
getdata1di64, 521
getdata1di8, 521
getdata2df32, 522
getdata2df64, 522
getdata2di16, 523
getdata2di32, 523
getdata2di64, 524
getdata2di8, 524
getdata3df32, 525
getdata3df64, 525
getdata3di16, 526
getdata3di32, 526
getdata3di64, 527
getdata3di8, 527
getdata4df32, 528
getdata4df64, 528
getdata4di16, 529
getdata4di32, 529
getdata4di64, 530
getdata4di8, 530
getdata5df32, 531
getdata5df64, 531
getdata5di16, 532
getdata5di32, 532
getdata5di64, 533
getdata5di8, 533
getdatascalarf32, 534
getdatascalarf64, 534
getdatascalari16, 535
getdatascalari32, 535
getdatascalari64, 536
getdatascalari8, 536
getdimensionbyid, 537
getdimensionbyname, 537
getdimensionlength, 537
getdimensionname, 538
getdtypefrominteger, 538
getdtypefromstring, 539

getglobalattributechar, 539
getglobalattributef32, 539
getglobalattributef64, 540
getglobalattributei16, 540
getglobalattributei32, 541
getglobalattributei64, 541
getglobalattributei8, 542
getnodimensions, 542
getnovariables, 542
getreaddatashape, 543
getunlimiteddimension, 544
getvariableattributechar, 545
getvariableattributef32, 545
getvariableattributef64, 546
getvariableattributei16, 546
getvariableattributei32, 546
getvariableattributei64, 547
getvariableattributei8, 547
getvariablebyname, 548
getvariabledimensions, 548
getvariabledtype, 548
getvariablefillvaluef32, 549
getvariablefillvaluef64, 549
getvariablefillvaluei16, 549
getvariablefillvaluei32, 549
getvariablefillvaluei64, 549
getvariablefillvaluei8, 549
getvariableids, 550
getvariablename, 550
getvariables, 550
getvariableshape, 550
hasattribute, 551
hasdimension, 551
hasvariable, 551
initncdataset, 551
initncdimension, 551
initncvariable, 552
isdatasetunlimited, 552
isunlimiteddimension, 552
isunlimitedvariable, 552
newncdataset, 552
newncdimension, 552
newncvariable, 553
setdata1df32, 553
setdata1df64, 553
setdata1di16, 554
setdata1di32, 554
setdata1di64, 555
setdata1di8, 555
setdata2df32, 556
setdata2df64, 556
setdata2di16, 557
setdata2di32, 557
setdata2di64, 558
setdata2di8, 558
setdata3df32, 559
setdata3df64, 559
setdata3di16, 560
setdata3di32, 560
setdata3di64, 561
setdata3di8, 561
setdata4df32, 562
setdata4df64, 562
setdata4di16, 563
setdata4di32, 563
setdata4di64, 564
setdata4di8, 564
setdata5df32, 565
setdata5df64, 565
setdata5di16, 566
setdata5di32, 566
setdata5di64, 567
setdata5di8, 567
setdatascalarf32, 568
setdatascalarf64, 568
setdatascalari16, 569
setdatascalari32, 569
setdatascalari64, 569
setdatascalari8, 570
setdimension, 570
setglobalattributechar, 571
setglobalattributef32, 571
setglobalattributef64, 571
setglobalattributei16, 572
setglobalattributei32, 572
setglobalattributei64, 573
setglobalattributei8, 573
setvariableattributechar, 573
setvariableattributef32, 574
setvariableattributef64, 574
setvariableattributei16, 575
setvariableattributei32, 575
setvariableattributei64, 575
setvariableattributei8, 576
setvariablefillvaluef32, 576
setvariablefillvaluef64, 576
setvariablefillvaluei16, 576
setvariablefillvaluei32, 577
setvariablefillvaluei64, 577
setvariablefillvaluei8, 577
setvariablewithids, 577
setvariablewithnames, 578
setvariablewithtypes, 579
mo_netcdf.f90, 1005
mo_netcdf::ncdataset, 836
close, 839
dataset, 846
file, 846
filename, 846
fname, 846
getattribute, 839
getdimension, 839
getdimensionbyid, 840
getdimensionbyname, 840
getglobalattributechar, 840
getglobalattributef32, 840

getglobalattributef64, 840
getglobalattributei16, 840
getglobalattributei32, 840
getglobalattributei64, 840
getglobalattributei8, 841
getnovariables, 841
getunlimiteddimension, 841
getvariable, 841
getvariablebyname, 842
getvariableids, 842
getvariables, 842
hasdimension, 842
hasvariable, 842
id, 846
initncdataset, 843
isunlimited, 843
mode, 847
netcdf, 847
of, 847
open, 847
opened, 847
setattribute, 843
setdimension, 844
setglobalattributechar, 844
setglobalattributef32, 844
setglobalattributef64, 844
setglobalattributei16, 845
setglobalattributei32, 845
setglobalattributei64, 845
setglobalattributei8, 845
setvariable, 845
setvariablewithids, 846
setvariablewithnames, 846
setvariablewithtypes, 846
the, 847
mo_netcdf::ncdimension, 847
dimension, 866
getattribute, 851
getdata, 852
getdata1df32, 852
getdata1df64, 852
getdata1di16, 852
getdata1di32, 852
getdata1di64, 852
getdata1di8, 853
getdata2df32, 853
getdata2df64, 853
getdata2di16, 853
getdata2di32, 853
getdata2di64, 853
getdata2di8, 853
getdata3df32, 853
getdata3df64, 853
getdata3di16, 854
getdata3di32, 854
getdata3di64, 854
getdata3di8, 854
getdata4df32, 854
getdata4df64, 854
getdata4di16, 854
getdata4di32, 854
getdata4di64, 854
getdata4di8, 855
getdata5df32, 855
getdata5df64, 855
getdata5di16, 855
getdata5di32, 855
getdata5di64, 855
getdata5di8, 855
getdatascalarf32, 855
getdatascalarf64, 855
getdatascalari16, 856
getdatascalari32, 856
getdatascalari64, 856
getdatascalari8, 856
getdimensions, 856
getdtype, 856
getfillvalue, 856
getname, 857
getnodimensions, 857
getshape, 857
getvariableattributechar, 857
getvariableattributef32, 857
getvariableattributef64, 857
getvariableattributei16, 857
getvariableattributei32, 857
getvariableattributei64, 858
getvariableattributei8, 858
getvariablefillvaluef32, 858
getvariablefillvaluef64, 858
getvariablefillvaluei16, 858
getvariablefillvaluei32, 858
getvariablefillvaluei64, 858
getvariablefillvaluei8, 858
hasattribute, 858
id, 866
initncvariable, 859
isunlimited, 859
netcdf, 866
parent, 866
s, 866
setattribute, 859
setdata, 859
setdata1df32, 860
setdata1df64, 860
setdata1di16, 860
setdata1di32, 860
setdata1di64, 860
setdata1di8, 860
setdata2df32, 861
setdata2df64, 861
setdata2di16, 861
setdata2di32, 861
setdata2di64, 861
setdata2di8, 861
setdata3df32, 861

setdata3df64, 861
setdata3di16, 861
setdata3di32, 862
setdata3di64, 862
setdata3di8, 862
setdata4df32, 862
setdata4df64, 862
setdata4di16, 862
setdata4di32, 862
setdata4di64, 862
setdata4di8, 862
setdata5df32, 863
setdata5df64, 863
setdata5di16, 863
setdata5di32, 863
setdata5di64, 863
setdata5di8, 863
setdatascalarf32, 863
setdatascalarf64, 863
setdatascalari16, 863
setdatascalari32, 864
setdatascalari64, 864
setdatascalari8, 864
setfillvalue, 864
setvariableattributechar, 864
setvariableattributef32, 864
setvariableattributef64, 865
setvariableattributei16, 865
setvariableattributei32, 865
setvariableattributei64, 865
setvariableattributei8, 865
setvariablefillvaluef32, 865
setvariablefillvaluef64, 865
setvariablefillvaluei16, 865
setvariablefillvaluei32, 865
setvariablefillvaluei64, 866
setvariablefillvaluei8, 866
the, 866, 867
mo_netcdf::ncvariable, 867
mo_neutrons, 579
approx_mon_int, 580
approx_mon_int_eps, 581
approx_mon_int_steps, 582
cosmic, 583
desiletsn0, 584
intgrandfast, 585
lookupintegral, 586
oldintegration, 586
tabularintegralafast, 587
mo_neutrons.f90, 1008
mo_nml, 588
close_nml, 589
length_error, 593
missing, 593
nunitnml, 593
open_nml, 590
position_nml, 591
positioned, 593
read_error, 593
mo_nml.f90, 1009
mo_objective_function, 594
extract_basin_avg_tws, 595
objective, 596
objective_et_kge_catchment_avg, 598
objective_kge_q_et, 599
objective_kge_q_rmse_et, 600
objective_kge_q_rmse_tws, 601
objective_kge_q_sm_corr, 603
objective_neutrons_kge_catchment_avg, 604
objective_sm_corr, 605
objective_sm_kge_catchment_avg, 606
objective_sm_pd, 608
objective_sm_sse_standard_score, 609
mo_objective_function.f90, 1009
mo_optimization, 610
optimization, 611
mo_optimization.f90, 1010
mo_optimization_utils, 612
mo_optimization_utils.f90, 1010
mo_optimization_utils::eval_interface, 783
eval_interface, 784
mo_optimization_utils::objective_interface, 875
objective_interface, 875
mo_orderpack, 613
d_ctrper, 615
d_fndnth, 615
d_indmed, 615
d_indnth, 616
d_inspar, 616
d_inssor, 616
d_med, 616
d_median, 617
d_mrgref, 617
d_mrgrnk, 617
d_mulcnt, 617
d_nearless, 617
d_rapknr, 617
d_refpar, 618
d_refsor, 618
d_rinpar, 618
d_rnkpar, 618
d_subsor, 618
d_uniinv, 619
d_unipar, 619
d_unirnk, 619
d_unista, 619
d_valmed, 619
d_valnth, 620
i_ctrper, 620
i_fndnth, 620
i_indmed, 620
i_indnth, 620
i_inspar, 621
i_inssor, 621
i_med, 621
i_median, 621

i_mrgref, 622
i_mrgrnk, 622
i_mulcnt, 622
i_nearless, 622
i_rapknr, 622
i_refpar, 622
i_refsor, 622
i_rinpar, 623
i_rnkpar, 623
i_subsor, 623
i_uniinv, 624
i_unipar, 624
i_unirnk, 624
i_unista, 624
i_valmed, 624
i_valnth, 624
idont, 630
r_ctrper, 625
r_fndnth, 625
r_indmed, 625
r_indnth, 625
r_inspar, 625
r_inssor, 626
r_med, 626
r_median, 626
r_mrgref, 626
r_mrgrnk, 627
r_mulcnt, 627
r_nearless, 627
r_rapknr, 627
r_refpar, 627
r_refsor, 627
r_rinpar, 628
r_rnkpar, 628
r_subsor, 628
r_uniinv, 628
r_unipar, 629
r_unirnk, 629
r_unista, 629
r_valmed, 629
r_valnth, 629
sort_index_dp, 629
sort_index_i4, 629
sort_index_sp, 630
mo_orderpack.f90, 1011
mo_orderpack::ctrper, 775
d_ctrper, 776
i_ctrper, 776
r_ctrper, 776
mo_orderpack::fndnth, 784
d_fndnth, 784
i_fndnth, 784
r_fndnth, 784
mo_orderpack::indmed, 802
d_indmed, 803
i_indmed, 803
r_indmed, 803
mo_orderpack::indnth, 803
d_indnth, 803
i_indnth, 803
r_indnth, 804
mo_orderpack::inspar, 804
d_inspar, 804
i_inspar, 804
r_inspar, 804
mo_orderpack::inssor, 805
d_inssor, 805
i_inssor, 805
r_inssor, 805
mo_orderpack::mrgref, 832
d_mrgref, 832
i_mrgref, 832
r_mrgref, 832
mo_orderpack::mrgrnk, 833
d_mrgrnk, 833
i_mrgrnk, 833
r_mrgrnk, 833
mo_orderpack::mulcnt, 835
d_mulcnt, 835
i_mulcnt, 835
r_mulcnt, 835
mo_orderpack::nearless, 868
d_nearless, 868
i_nearless, 868
r_nearless, 868
mo_orderpack::omedian, 875
d_median, 875
i_median, 875
r_median, 875
mo_orderpack::rapknr, 902
d_rapknr, 902
i_rapknr, 903
r_rapknr, 903
mo_orderpack::refpar, 906
d_refpar, 906
i_refpar, 906
r_refpar, 906
mo_orderpack::refsor, 907
d_refsor, 907
i_refsor, 907
r_refsor, 907
mo_orderpack::rinpar, 907
d_rinpar, 907
i_rinpar, 908
r_rinpar, 908
mo_orderpack::rnkpar, 909
d_rnkpar, 909
i_rnkpar, 910
r_rnkpar, 910
mo_orderpack::sort, 916
d_refsor, 919
i_refsor, 919
r_refsor, 919
mo_orderpack::sort_index, 919
sort_index_dp, 919
sort_index_i4, 920

sort_index_sp, 920
mo_orderpack::uniinv, 933
d_uniinv, 933
i_uniinv, 933
r_uniinv, 933
mo_orderpack::unipar, 934
d_unipar, 934
i_unipar, 934
r_unipar, 934
mo_orderpack::unirnk, 934
d_unirnk, 934
i_unirnk, 935
r_unirnk, 935
mo_orderpack::unista, 935
d_unista, 935
i_unista, 935
r_unista, 935
mo_orderpack::valmed, 939
d_valmed, 939
i_valmed, 939
r_valmed, 940
mo_orderpack::valnth, 940
d_valnth, 940
i_valnth, 940
r_valnth, 940
mo_percentile, 630
median_dp, 630
median_sp, 631
n_element_dp, 631
n_element_sp, 631
percentile_0d_dp, 631
percentile_0d_sp, 631
percentile_1d_dp, 632
percentile_1d_sp, 632
qmedian_dp, 632
qmedian_sp, 632
mo_percentile.f90, 1013
mo_percentile::median, 829
median_dp, 829
median_sp, 830
mo_percentile::n_element, 835
n_element_dp, 836
n_element_sp, 836
mo_percentile::percentile, 899
percentile_0d_dp, 899
percentile_0d_sp, 899
percentile_1d_dp, 899
percentile_1d_sp, 900
mo_percentile::qmedian, 902
qmedian_dp, 902
qmedian_sp, 902
mo_pet, 632
extraterr_rad_approx, 633
pet_hargreaves, 634
pet_penman, 635
pet_priestly, 637
sat_vap_pressure, 638
slope_satpressure, 639
mo_pet.f90, 1013
mo_prepare_gridded_lai, 640
prepare_gridded_daily_lai_data, 641
prepare_gridded_mean_monthly_lai_data, 642
mo_prepare_gridded_lai.f90, 1014
mo_read_forcing_nc, 643
get_time_vector_and_select, 644
read_const_forcing_nc, 645
read_forcing_nc, 646
read_weights_nc, 648
mo_read_forcing_nc.f90, 1014
mo_read_latlon, 649
read_latlon, 650
mo_read_latlon.f90, 1015
mo_read_lut, 651
read_geoformation_lut, 652
read_lai_lut, 653
mo_read_lut.f90, 1015
mo_read_optional_data, 654
read_basin_avg_tws, 654
read_evapotranspiration, 655
read_neutrons, 656
read_soil_moisture, 657
mo_read_optional_data.f90, 1015
mo_read_spatial_data, 658
read_header_ascii, 659
read_spatial_data_ascii_dp, 660
read_spatial_data_ascii_i4, 661
mo_read_spatial_data.f90, 1016
mo_read_spatial_data::read_spatial_data_ascii, 903
read_spatial_data_ascii_dp, 904
read_spatial_data_ascii_i4, 904
mo_read_timeseries, 662
read_timeseries, 662
mo_read_timeseries.f90, 1016
mo_read_wrapper, 664
check_consistency_lut_map, 665
read_data, 666
mo_read_wrapper.f90, 1016
mo_restart, 667
read_restart_states, 668
unpack_field_and_write_1d_dp, 669
unpack_field_and_write_1d_i4, 669
unpack_field_and_write_2d_dp, 670
unpack_field_and_write_3d_dp, 670
write_restart_files, 670
mo_restart.f90, 1017
mo_restart::unpack_field_and_write, 936
unpack_field_and_write_1d_dp, 936
unpack_field_and_write_1d_i4, 937
unpack_field_and_write_2d_dp, 937
unpack_field_and_write_3d_dp, 937
mo_runoff, 671
l1_total_runoff, 672
runoff_sat_zone, 673
runoff_unsat_zone, 673
mo_runoff.f90, 1017
mo_sce, 675

cce, 675
chkcst, 676
comp, 676
getpnt, 677
parstt, 678
sce, 678
sort_matrix, 682
mo_sce.f90, 1017
set_optional, 1018
write_best_final, 1018
write_best_intermediate, 1019
write_population, 1019
write_termination_case, 1019
mo_set_netcdf_outputs, 683
set_netcdf, 683
mo_set_netcdf_outputs.f90, 1020
mo_snow_accum_melt, 684
snow_accum_melt, 684
mo_snow_accum_melt.f90, 1020
mo_soil_database, 685
generate_soil_database, 686
read_soil_lut, 687
mo_soil_database.f90, 1020
mo_soil_moisture, 688
feddes_et_reduction, 688
jarvis_et_reduction, 689
soil_moisture, 690
mo_soil_moisture.f90, 1021
mo_spatial_agg_disagg_forcing, 692
spatial_aggregation_3d, 693
spatial_aggregation_4d, 693
spatial_disaggregation_3d, 693
spatial_disaggregation_4d, 694
mo_spatial_agg_disagg_forcing.f90, 1021
mo_spatial_agg_disagg_forcing::spatial_aggregation, 920
spatial_aggregation_3d, 920
spatial_aggregation_4d, 921
mo_spatial_agg_disagg_forcing::spatial_disaggregation, 921
spatial_disaggregation_3d, 922
spatial_disaggregation_4d, 922
mo_spatialsimilarity, 694
nndv_dp, 695
nndv_sp, 695
pd_dp, 695
pd_sp, 695
mo_spatialsimilarity.f90, 1022
mo_spatialsimilarity::nndv, 868
nndv_dp, 870
nndv_sp, 870
mo_spatialsimilarity::pd, 897
pd_dp, 898
pd_sp, 899
mo_standard_score, 695
classified_standard_score_dp, 696
classified_standard_score_sp, 696
standard_score_dp, 696
standard_score_sp, 697
mo_standard_score.f90, 1022
mo_standard_score::classified_standard_score, 771
classified_standard_score_dp, 772
classified_standard_score_sp, 772
mo_standard_score::standard_score, 928
standard_score_dp, 929
standard_score_sp, 929
mo_startup, 697
constants_init, 697
l2_variable_init, 698
mhm_initialize, 699
mo_startup.f90, 1022
mo_string_utils, 701
compress, 701
divide_string, 702
dp2str, 703
equalstrings, 703
i42str, 703
i4array2str, 704
i82str, 704
log2str, 704
nonull, 704
separator, 707
sp2str, 705
splitstring, 705
startswith, 705
str2num, 706
tolower, 706
toupper, 707
mo_string_utils.f90, 1023
mo_string_utils::num2str, 872
dp2str, 873
i42str, 873
i82str, 873
log2str, 873
sp2str, 873
mo_string_utils::numarray2str, 874
i4array2str, 874
mo_template, 708
circum, 708
itest, 709
mean_dp, 709
mean_sp, 709
pi_dp, 709
pi_sp, 710
mo_template.f90, 1024
mo_template::mean, 828
mean_dp, 829
mean_sp, 829
mo_temporal_aggregation, 710
day2mon_average_dp, 710
hour2day_average_dp, 711
mo_temporal_aggregation.f90, 1024
mo_temporal_aggregation::day2mon_average, 776
day2mon_average_dp, 777
mo_temporal_aggregation::hour2day_average, 801
hour2day_average_dp, 802

mo_temporal_disagg_forcing, 711
temporal_disagg_forcing, 712
mo_temporal_disagg_forcing.f90, 1025
mo_timer, 713
clock_rate, 721
cputime, 721
cycles1, 721
cycles2, 721
cycles_max, 722
max_timers, 722
status, 722
timer_check, 714
timer_clear, 715
timer_get, 716
timer_print, 717
timer_start, 718
timer_stop, 719
timers_init, 720
mo_timer.f90, 1025
mo_upscaling_operators, 722
l0_fractionalcover_in_lx, 723
majority_statistics, 724
upscale_arithmetic_mean, 724
upscale_geometric_mean, 726
upscale_harmonic_mean, 727
upscale_p_norm, 728
mo_upscaling_operators.f90, 1026
mo_utils, 729
equal_dp, 730
equal_sp, 730
greaterequal_dp, 730
greaterequal_sp, 730
is_finite_dp, 731
is_finite_sp, 731
is_nan_dp, 731
is_nan_sp, 731
is_normal_dp, 731
is_normal_sp, 731
lesserequal_dp, 731
lesserequal_sp, 731
locate_0d_dp, 732
locate_0d_sp, 732
locate_1d_dp, 732
locate_1d_sp, 732
notequal_dp, 732
notequal_sp, 732
special_value_dp, 732
special_value_sp, 733
swap_vec_dp, 733
swap_vec_i4, 733
swap_vec_sp, 734
swap_xy_dp, 734
swap_xy_i4, 734
swap_xy_sp, 734
mo_utils.f90, 1026
mo_utils::eq, 782
equal_dp, 782
equal_sp, 782
mo_utils::equal, 782
equal_dp, 783
equal_sp, 783
mo_utils::ge, 787
greaterequal_dp, 787
greaterequal_sp, 787
mo_utils::greaterequal, 796
greaterequal_dp, 796
greaterequal_sp, 796
mo_utils::is_finite, 805
is_finite_dp, 806
is_finite_sp, 806
mo_utils::is_nan, 806
is_nan_dp, 806
is_nan_sp, 806
mo_utils::is_normal, 807
is_normal_dp, 807
is_normal_sp, 807
mo_utils::le, 812
lesserequal_dp, 812
lesserequal_sp, 812
mo_utils::lesserequal, 813
lesserequal_dp, 813
lesserequal_sp, 813
mo_utils::locate, 816
locate_0d_dp, 816
locate_0d_sp, 816
locate_1d_dp, 817
locate_1d_sp, 817
mo_utils::ne, 867
notequal_dp, 867
notequal_sp, 867
mo_utils::notequal, 870
notequal_dp, 870
notequal_sp, 870
mo_utils::special_value, 922
special_value_dp, 923
special_value_sp, 923
mo_utils::swap, 930
swap_vec_dp, 931
swap_vec_i4, 931
swap_vec_sp, 931
swap_xy_dp, 931
swap_xy_i4, 931
swap_xy_sp, 932
mo_write_ascii, 734
write_configfile, 735
write_optifile, 736
write_optinamelist, 737
mo_write_ascii.f90, 1027
mo_write_fluxes_states, 738
close, 739
createoutputfile, 739
fluxesunit, 740
newoutputdataset, 741
newoutputvariable, 742
updatedataset, 743
updatevariable, 744

writetimestep, 745
writevariableattributes, 746
writevariabletimestep, 746
mo_write_fluxes_states.f90, 1028
mo_write_fluxes_states::outputdataset, 880
all, 881
basin, 881
close, 881
count, 881
counter, 882
created, 882
ibasin, 882
id, 882
nc, 882
ncdataset, 882
steps, 882
store, 882
time, 882
to, 883
updatedataset, 881
variables, 883
vars, 883
write, 883
writetimestep, 881
written, 883
mo_write_fluxes_states::outputvariable, 889
average, 890
avg, 890
before, 890
between, 891
calls, 891
contains, 891
count, 891
counter, 891
data, 891
mask, 891
nc, 891
ncdataset, 892
number, 892
of, 892
reconstruct, 892
store, 892
the, 892
to, 892
updatevariable, 890, 893
variable, 893
which, 893
writes, 893
writevariabletimestep, 890
writing, 893
mo_xor4096, 747
get_timeseed_i4_0d, 748
get_timeseed_i4_1d, 748
get_timeseed_i8_0d, 748
get_timeseed_i8_1d, 748
n_save_state, 751
xor4096d_0d, 748
xor4096d_1d, 748
xor4096f_0d, 749
xor4096f_1d, 749
xor4096gd_0d, 749
xor4096gd_1d, 749
xor4096gf_0d, 749
xor4096gf_1d, 750
xor4096l_0d, 750
xor4096l_1d, 750
xor4096s_0d, 750
xor4096s_1d, 750
mo_xor4096.f90, 1029
mo_xor4096::get_timeseed, 793
get_timeseed_i4_0d, 793
get_timeseed_i4_1d, 794
get_timeseed_i8_0d, 794
get_timeseed_i8_1d, 794
mo_xor4096::xor4096, 955
xor4096d_0d, 955
xor4096d_1d, 955
xor4096f_0d, 955
xor4096f_1d, 956
xor4096l_0d, 956
xor4096l_1d, 956
xor4096s_0d, 956
xor4096s_1d, 956
mo_xor4096::xor4096g, 957
xor4096gd_0d, 957
xor4096gd_1d, 957
xor4096gf_0d, 957
xor4096gf_1d, 957
mode
mo_netcdf::ncdataset, 847
moment_dp
mo_moment, 269
mo_moment::moment, 831
moment_sp
mo_moment, 269
mo_moment::moment, 832
moments
mo_mrm_signatures, 432
month_counter
mo_mrm_write, 446
movedownonecell
mo_mrm_net_startup, 372
moveup
mo_mrm_net_startup, 373
mpr
mo_multi_param_reg, 461
mpr_driver
mpr_driver.f90, 1029
mpr_driver.f90, 1029
mpr_driver, 1029
mpr_eval
mo_mpr_eval, 277
mpr_initialize
mo_mpr_startup, 322
mpr_read_config
mo_mpr_read_config, 300

mpr_runoff
mo_mpr_runoff, 307
mpr_sm
mo_mpr_soilmoist, 315
mpr_smhorizons
mo_mpr_smhorizons, 309
mrm_coupling_mode
mo_common_mhm_mrm_variables, 110
mrm_driver
mrm_driver.f90, 1032
mrm_driver.f90, 1031
mrm_driver, 1032
mrm_eval
mo_mrm_eval, 325
mrm_init
mo_mrm_init, 348
mrm_init_param
mo_mrm_mpr, 354
mrm_read_bankfull_runoff
mo_mrm_read_data, 404
mrm_read_config
mo_mrm_read_config, 401
mrm_read_discharge
mo_mrm_read_data, 405
mrm_read_l0_data
mo_mrm_read_data, 406
mrm_read_restart_config
mo_mrm_restart, 410
mrm_read_restart_states
mo_mrm_restart, 411
mrm_read_total_runoff
mo_mrm_read_data, 407
mrm_routing
mo_mrm_routing, 424
mrm_runoff
mo_mrm_global_variables, 344
mrm_update_param
mo_mrm_mpr, 355
mrm_write
mo_mrm_write, 437
mrm_write_optifile
mo_mrm_write, 438
mrm_write_optinamelist
mo_mrm_write, 439
mrm_write_output_fluxes
mo_mrm_write, 441
mrm_write_restart
mo_mrm_restart, 412
mse_dp_1d
mo_errormeasures, 158
mo_errormeasures::mse, 834
mse_dp_2d
mo_errormeasures, 159
mo_errormeasures::mse, 834
mse_dp_3d
mo_errormeasures, 160
mo_errormeasures::mse, 834
mse_sp_1d
mo_errormeasures, 160
mo_errormeasures::mse, 834
mse_sp_2d
mo_errormeasures, 161
mo_errormeasures::mse, 834
mse_sp_3d
mo_errormeasures, 162
mo_errormeasures::mse, 834
mstart
mo_common_variables::period, 901
multi_objective_ae_fdc_lsv_nse_djf
mo_mrm_objective_function_runoff, 381
multi_objective_lnnse_highflow_lnnse_lowflow
mo_mrm_objective_function_runoff, 382
multi_objective_lnnse_highflow_lnnse_lowflow_2
mo_mrm_objective_function_runoff, 384
multi_objective_nse_lnnse
mo_mrm_objective_function_runoff, 385
multi_objective_runoff
mo_mrm_objective_function_runoff, 386

n
mo_kind::sprs2_dp, 925
mo_kind::sprs2_sp, 926
n_element_dp
mo_percentile, 631
mo_percentile::n_element, 836
n_element_sp
mo_percentile, 631
mo_percentile::n_element, 836
n_save_state
mo_xor4096, 751
n_subcells
mo_common_variables::gridremapper, 801
name
mo_ncwrite::attribute, 763
mo_ncwrite::dims, 778
mo_ncwrite::variable, 952
natt
mo_ncwrite::variable, 952
nattdim
mo_ncwrite, 514
nbasins
mo_common_variables, 128
nc
mo_mrm_write, 446
mo_mrm_write_fluxes_states::outputdataset, 878
mo_mrm_write_fluxes_states::outputvariable, 886
mo_write_fluxes_states::outputdataset, 882
mo_write_fluxes_states::outputvariable, 891
nc_riverhead
mo_mrm_river_head, 418
nc_time
mo_mrm_river_head, 419
ncclose
mo_ncread, 486
ncdataset
mo_mrm_write_fluxes_states::outputdataset, 878
mo_mrm_write_fluxes_states::outputvariable, 887

mo_write_fluxes_states::outputdataset, 882
mo_write_fluxes_states::outputvariable, 892
ncells
mo_common_variables::grid, 798
ncfile_discharge
mo_mrm_file, 331
ncolpars
mo_common_constants, 99
ncols
mo_common_variables::grid, 798
ncopen
mo_ncread, 487
ndays
mo_julian, 224
ndims
mo_ncwrite, 514
mo_ncwrite::variable, 952
ndyin
mo_julian, 225
neigh_value
mo_dds, 150
nerr
mo_constants, 133
nerror_model
mo_common_mhm_mrm_variables, 111
netcdf
mo_netcdf::ncdataset, 847
mo_netcdf::ncdimension, 866
neutron_integral_afast
mo_global_variables, 190
newncdataset
mo_netcdf, 552
newncdimension
mo_netcdf, 552
newncvariable
mo_netcdf, 553
newoutputdataset
mo_mrm_write_fluxes_states, 449
mo_write_fluxes_states, 741
newoutputvariable
mo_mrm_write_fluxes_states, 450
mo_write_fluxes_states, 742
ngatt
mo_ncwrite, 514
ngauges
mo_mrm_global_variables::basininfo_mrm, 768
ngaugestotal
mo_mrm_global_variables, 344
ngeounits
mo_mpr_global_variables, 293
nhorizons
mo_mpr_global_variables::soiltype, 915
nin
mo_constants, 133
ninflowgauges
mo_mrm_global_variables::basininfo_mrm, 768
ninflowgaugestotal
mo_mrm_global_variables, 344
niterations
mo_common_mhm_mrm_variables, 111
nlai
mo_mpr_global_variables, 293
nlaiclass
mo_mpr_global_variables, 293
nlcover_class
mo_mpr_constants, 273
nlcoverscene
mo_common_variables, 128
nlvls
mo_ncwrite::variable, 952
nmaxatt
mo_ncwrite, 514
nmaxdim
mo_ncwrite, 514
nmeasperday
mo_mrm_global_variables, 345
nmeasperday_tws
mo_global_variables, 190
nndv_dp
mo_spatialsimilarity, 695
mo_spatialsimilarity::nndv, 870
nndv_sp
mo_spatialsimilarity, 695
mo_spatialsimilarity::nndv, 870
nnml
mo_constants, 134
nobs
mo_common_variables::period, 901
nodata_dp
mo_common_constants, 99
nodata_i4
mo_common_constants, 100
nodata_value
mo_common_variables::grid, 798
nonull
mo_string_utils, 704
notequal_dp
mo_utils, 732
mo_utils::ne, 867
mo_utils::notequal, 870
notequal_sp
mo_utils, 732
mo_utils::ne, 867
mo_utils::notequal, 870
nout
mo_constants, 134
noutflxstate
mo_mhm_constants, 257
mo_mrm_constants, 324
npar2_arth
mo_corr, 146
npar_arth
mo_corr, 146
nprocesses
mo_common_variables, 128
nroutingstates

mo_mrm_constants, 324
nrows
mo_common_variables::grid, 798
nse_dp_1d
mo_errormeasures, 162
mo_errormeasures::nse, 871
nse_dp_2d
mo_errormeasures, 163
mo_errormeasures::nse, 871
nse_dp_3d
mo_errormeasures, 163
mo_errormeasures::nse, 871
nse_sp_1d
mo_errormeasures, 163
mo_errormeasures::nse, 871
nse_sp_2d
mo_errormeasures, 163
mo_errormeasures::nse, 872
nse_sp_3d
mo_errormeasures, 163
mo_errormeasures::nse, 872
nsoilhorizons_mhm
mo_mpr_global_variables, 294
nsoilhorizons_sm_input
mo_global_variables, 190
nsoiltypes
mo_mpr_global_variables, 294
nsubs
mo_ncwrite::variable, 952
ntillhorizons
mo_mpr_global_variables::soiltype, 915
ntimesteps_l1_et
mo_global_variables, 190
ntimesteps_l1_neutrons
mo_global_variables, 190
ntimesteps_l1_sm
mo_global_variables, 190
ntstepday
mo_common_mhm_mrm_variables, 111
mo_mrm_global_variables, 345
number
mo_mrm_write_fluxes_states::outputvariable, 887
mo_write_fluxes_states::outputvariable, 892
nuniquel0basins
mo_common_variables, 128
nunitnml
mo_nml, 593
nvalues
mo_ncwrite::attribute, 763
nvars
mo_ncwrite, 514

objective
mo_objective_function, 596
objective_equal_nse_lnnse
mo_mrm_objective_function_runoff, 387
objective_et_kge_catchment_avg
mo_objective_function, 598
objective_interface
mo_optimization_utils::objective_interface, 875
objective_kge
mo_mrm_objective_function_runoff, 389
objective_kge_q_et
mo_objective_function, 599
objective_kge_q_rmse_et
mo_objective_function, 600
objective_kge_q_rmse_tws
mo_objective_function, 601
objective_kge_q_sm_corr
mo_objective_function, 603
objective_lnnse
mo_mrm_objective_function_runoff, 390
objective_multiple_gauges_kge_power6
mo_mrm_objective_function_runoff, 392
objective_neutrons_kge_catchment_avg
mo_objective_function, 604
objective_nse
mo_mrm_objective_function_runoff, 393
objective_power6_nse_lnnse
mo_mrm_objective_function_runoff, 394
objective_sm_corr
mo_objective_function, 605
objective_sm_kge_catchment_avg
mo_objective_function, 606
objective_sm_pd
mo_objective_function, 608
objective_sm_sse_standard_score
mo_objective_function, 609
objective_sse
mo_mrm_objective_function_runoff, 395
objective_weighted_nse
mo_mrm_objective_function_runoff, 397
of
mo_mrm_write_fluxes_states::outputvariable, 887
mo_netcdf::ncdataset, 847
mo_write_fluxes_states::outputvariable, 892
oldintegration
mo_neutrons, 586
open
mo_netcdf::ncdataset, 847
open_netcdf
mo_ncwrite, 500
open_nml
mo_nml, 590
opened
mo_netcdf::ncdataset, 847
opti_function
mo_common_mhm_mrm_variables, 111
opti_method
mo_common_mhm_mrm_variables, 111
optimization
mo_optimization, 611
optimize
mo_common_mhm_mrm_variables, 111
optimize_restart
mo_common_mhm_mrm_variables, 112
outputflxstate

mo_global_variables, 190
outputflxstate_mrm
mo_mrm_global_variables, 345

p0_dp
mo_constants, 134
p0_sp
mo_constants, 134
p1_initstatefluxes
mo_common_constants, 100
p2_initstatefluxes
mo_mhm_constants, 257
mo_mpr_constants, 273
p3_initstatefluxes
mo_mhm_constants, 257
mo_mpr_constants, 273
p4_initstatefluxes
mo_mhm_constants, 257
mo_mpr_constants, 273
p5_initstatefluxes
mo_mhm_constants, 258
mo_mpr_constants, 274
parameter_regularization
mo_mrm_objective_function_runoff, 398
parent
mo_netcdf::ncdimension, 866
pargen_anneal_dp
mo_anneal, 86
pargen_dds_dp
mo_anneal, 86
pargen_dp
mo_mcmc, 237
pargennorm_dp
mo_mcmc, 237
parstt
mo_sce, 678
paste_char_m_m
mo_append, 92
mo_append::paste, 894
paste_char_m_s
mo_append, 92
mo_append::paste, 894
paste_char_m_v
mo_append, 92
mo_append::paste, 895
paste_dp_m_m
mo_append, 92
mo_append::paste, 895
paste_dp_m_s
mo_append, 93
mo_append::paste, 895
paste_dp_m_v
mo_append, 93
mo_append::paste, 895
paste_i4_m_m
mo_append, 93
mo_append::paste, 895
paste_i4_m_s
mo_append, 93
mo_append::paste, 895
paste_i4_m_v
mo_append, 93
mo_append::paste, 895
paste_i8_m_m
mo_append, 93
mo_append::paste, 896
paste_i8_m_s
mo_append, 93
mo_append::paste, 896
paste_i8_m_v
mo_append, 94
mo_append::paste, 896
paste_lgt_m_m
mo_append, 94
mo_append::paste, 896
paste_lgt_m_s
mo_append, 94
mo_append::paste, 896
paste_lgt_m_v
mo_append, 94
mo_append::paste, 896
paste_sp_m_m
mo_append, 94
mo_append::paste, 897
paste_sp_m_s
mo_append, 94
mo_append::paste, 897
paste_sp_m_v
mo_append, 94
mo_append::paste, 897
pd_dp
mo_spatialsimilarity, 695
mo_spatialsimilarity::pd, 898
pd_sp
mo_spatialsimilarity, 695
mo_spatialsimilarity::pd, 899
peakdistribution
mo_mrm_signatures, 433
percentile_0d_dp
mo_percentile, 631
mo_percentile::percentile, 899
percentile_0d_sp
mo_percentile, 631
mo_percentile::percentile, 899
percentile_1d_dp
mo_percentile, 632
mo_percentile::percentile, 899
percentile_1d_sp
mo_percentile, 632
mo_percentile::percentile, 900
pet_correctbyasp
mo_mpr_pet, 296
pet_correctbylai
mo_mpr_pet, 297
pet_hargreaves
mo_pet, 634
pet_penman

mo_pet, 635
pet_priestly
mo_pet, 637
pi
mo_constants, 134
pi_d
mo_constants, 134
pi_dp
mo_constants, 134
mo_template, 709
pi_sp
mo_constants, 135
mo_template, 710
pio2
mo_constants, 135
pio2_d
mo_constants, 135
pio2_dp
mo_constants, 135
pio2_sp
mo_constants, 135
position_nml
mo_nml, 591
positioned
mo_nml, 593
prepare_gridded_daily_lai_data
mo_prepare_gridded_lai, 641
prepare_gridded_mean_monthly_lai_data
mo_prepare_gridded_lai, 642
prepare_meteo_forcings_data
mo_meteo_forcings, 247
priestley_taylor_alpha
mo_mpr_pet, 299
print_startup_message
mo_mrm_init, 351
processmatrix
mo_common_variables, 129
project_details
mo_common_variables, 129
psychro_dp
mo_constants, 135
psychro_sp
mo_constants, 135
pwp
mo_mpr_soilmoist, 317
pwp_c
mo_mpr_constants, 274
pwp_matpot_thetar
mo_mpr_constants, 274

q
mo_mrm_global_variables::gaugingstation, 787
qmedian_dp
mo_percentile, 632
mo_percentile::qmedian, 902
qmedian_sp
mo_percentile, 632
mo_percentile::qmedian, 902

r_ctrper
mo_orderpack, 625
mo_orderpack::ctrper, 776
r_fndnth
mo_orderpack, 625
mo_orderpack::fndnth, 784
r_indmed
mo_orderpack, 625
mo_orderpack::indmed, 803
r_indnth
mo_orderpack, 625
mo_orderpack::indnth, 804
r_inspar
mo_orderpack, 625
mo_orderpack::inspar, 804
r_inssor
mo_orderpack, 626
mo_orderpack::inssor, 805
r_med
mo_orderpack, 626
r_median
mo_orderpack, 626
mo_orderpack::omedian, 875
r_mrgref
mo_orderpack, 626
mo_orderpack::mrgref, 832
r_mrgrnk
mo_orderpack, 627
mo_orderpack::mrgrnk, 833
r_mulcnt
mo_orderpack, 627
mo_orderpack::mulcnt, 835
r_nearless
mo_orderpack, 627
mo_orderpack::nearless, 868
r_rapknr
mo_orderpack, 627
mo_orderpack::rapknr, 903
r_refpar
mo_orderpack, 627
mo_orderpack::refpar, 906
r_refsor
mo_orderpack, 627
mo_orderpack::refsor, 907
mo_orderpack::sort, 919
r_rinpar
mo_orderpack, 628
mo_orderpack::rinpar, 908
r_rnkpar
mo_orderpack, 628
mo_orderpack::rnkpar, 910
r_subsor
mo_orderpack, 628
r_uniinv
mo_orderpack, 628
mo_orderpack::uniinv, 933
r_unipar
mo_orderpack, 629

mo_orderpack::unipar, 934
r_unirnk
mo_orderpack, 629
mo_orderpack::unirnk, 935
r_unista
mo_orderpack, 629
mo_orderpack::unista, 935
r_valmed
mo_orderpack, 629
mo_orderpack::valmed, 940
r_valnth
mo_orderpack, 629
mo_orderpack::valnth, 940
RELEASES.md, 1033
rad2deg_dp
mo_constants, 136
rad2deg_sp
mo_constants, 136
radiusearth_dp
mo_constants, 136
radiusearth_sp
mo_constants, 136
read_basin_avg_tws
mo_read_optional_data, 654
read_const_forcing_nc
mo_read_forcing_nc, 645
read_data
mo_read_wrapper, 666
read_dem
mo_common_read_data, 117
read_error
mo_nml, 593
read_evapotranspiration
mo_read_optional_data, 655
read_forcing_nc
mo_read_forcing_nc, 646
read_geoformation_lut
mo_read_lut, 652
read_grid_info
mo_common_restart, 119
read_header_ascii
mo_read_spatial_data, 659
read_lai_lut
mo_read_lut, 653
read_latlon
mo_read_latlon, 650
read_lcover
mo_common_read_data, 118
read_meteo_weights
mo_global_variables, 190
read_mrm_routing_params
mo_mrm_read_config, 402
read_neutrons
mo_read_optional_data, 656
read_restart
mo_common_mhm_mrm_variables, 112
read_restart_states
mo_restart, 668
read_soil_lut
mo_soil_database, 687
read_soil_moisture
mo_read_optional_data, 657
read_spatial_data_ascii_dp
mo_read_spatial_data, 660
mo_read_spatial_data::read_spatial_data_ascii, 904
read_spatial_data_ascii_i4
mo_read_spatial_data, 661
mo_read_spatial_data::read_spatial_data_ascii, 904
read_timeseries
mo_read_timeseries, 662
read_weights_nc
mo_read_forcing_nc, 648
readper
mo_common_mhm_mrm_variables, 112
realft_dp
mo_corr, 144
mo_corr::realft, 905
realft_sp
mo_corr, 144
mo_corr::realft, 905
reconstruct
mo_mrm_write_fluxes_states::outputvariable, 887
mo_write_fluxes_states::outputvariable, 892
reg_rout
mo_mrm_mpr, 356
reset_sum
mo_mrm_river_head, 418
resolutionhydrology
mo_common_variables, 129
resolutionrouting
mo_common_mhm_mrm_variables, 112
rho0_dp
mo_constants, 136
rho0_sp
mo_constants, 136
right_bound
mo_common_variables::gridremapper, 801
rmse_dp_1d
mo_errormeasures, 163
mo_errormeasures::rmse, 908
rmse_dp_2d
mo_errormeasures, 164
mo_errormeasures::rmse, 908
rmse_dp_3d
mo_errormeasures, 164
mo_errormeasures::rmse, 909
rmse_sp_1d
mo_errormeasures, 164
mo_errormeasures::rmse, 909
rmse_sp_2d
mo_errormeasures, 165
mo_errormeasures::rmse, 909
rmse_sp_3d
mo_errormeasures, 165

mo_errormeasures::rmse, 909
rotate_fdir_variable
mo_mrm_read_data, 408
rout_space_weight
mo_mrm_constants, 325
routingstates
mo_global_variables, 191
runoff_sat_zone
mo_runoff, 673
runoff_unsat_zone
mo_runoff, 673
runoffratio
mo_mrm_signatures, 434
rzdepth
mo_mpr_global_variables::soiltype, 915

s
mo_netcdf::ncdimension, 866
sa_temp
mo_common_mhm_mrm_variables, 112
sae_dp_1d
mo_errormeasures, 166
mo_errormeasures::sae, 910
sae_dp_2d
mo_errormeasures, 166
mo_errormeasures::sae, 910
sae_dp_3d
mo_errormeasures, 167
mo_errormeasures::sae, 911
sae_sp_1d
mo_errormeasures, 168
mo_errormeasures::sae, 911
sae_sp_2d
mo_errormeasures, 168
mo_errormeasures::sae, 911
sae_sp_3d
mo_errormeasures, 169
mo_errormeasures::sae, 911
sand
mo_mpr_global_variables::soiltype, 915
sat_vap_pressure
mo_pet, 638
satpressureslope1
mo_mhm_constants, 258
sce
mo_sce, 678
sce_ngs
mo_common_mhm_mrm_variables, 112
sce_npg
mo_common_mhm_mrm_variables, 113
sce_nps
mo_common_mhm_mrm_variables, 113
secday_dp
mo_constants, 136
secday_sp
mo_constants, 136
seed
mo_common_mhm_mrm_variables, 113
selectcalendar
mo_julian, 226
separator
mo_string_utils, 707
set_basin_indices
mo_grid, 197
set_land_cover_scenes_id
mo_common_read_config, 116
set_netcdf
mo_set_netcdf_outputs, 683
set_optional
mo_sce.f90, 1018
setattribute
mo_netcdf::ncdataset, 843
mo_netcdf::ncdimension, 859
setcalendarinteger
mo_julian, 227
mo_julian::setcalendar, 912
setcalendarstring
mo_julian, 228
mo_julian::setcalendar, 912
setdata
mo_netcdf::ncdimension, 859
setdata1df32
mo_netcdf, 553
mo_netcdf::ncdimension, 860
setdata1df64
mo_netcdf, 553
mo_netcdf::ncdimension, 860
setdata1di16
mo_netcdf, 554
mo_netcdf::ncdimension, 860
setdata1di32
mo_netcdf, 554
mo_netcdf::ncdimension, 860
setdata1di64
mo_netcdf, 555
mo_netcdf::ncdimension, 860
setdata1di8
mo_netcdf, 555
mo_netcdf::ncdimension, 860
setdata2df32
mo_netcdf, 556
mo_netcdf::ncdimension, 861
setdata2df64
mo_netcdf, 556
mo_netcdf::ncdimension, 861
setdata2di16
mo_netcdf, 557
mo_netcdf::ncdimension, 861
setdata2di32
mo_netcdf, 557
mo_netcdf::ncdimension, 861
setdata2di64
mo_netcdf, 558
mo_netcdf::ncdimension, 861
setdata2di8
mo_netcdf, 558
mo_netcdf::ncdimension, 861

setdata3df32
mo_netcdf, 559
mo_netcdf::ncdimension, 861
setdata3df64
mo_netcdf, 559
mo_netcdf::ncdimension, 861
setdata3di16
mo_netcdf, 560
mo_netcdf::ncdimension, 861
setdata3di32
mo_netcdf, 560
mo_netcdf::ncdimension, 862
setdata3di64
mo_netcdf, 561
mo_netcdf::ncdimension, 862
setdata3di8
mo_netcdf, 561
mo_netcdf::ncdimension, 862
setdata4df32
mo_netcdf, 562
mo_netcdf::ncdimension, 862
setdata4df64
mo_netcdf, 562
mo_netcdf::ncdimension, 862
setdata4di16
mo_netcdf, 563
mo_netcdf::ncdimension, 862
setdata4di32
mo_netcdf, 563
mo_netcdf::ncdimension, 862
setdata4di64
mo_netcdf, 564
mo_netcdf::ncdimension, 862
setdata4di8
mo_netcdf, 564
mo_netcdf::ncdimension, 862
setdata5df32
mo_netcdf, 565
mo_netcdf::ncdimension, 863
setdata5df64
mo_netcdf, 565
mo_netcdf::ncdimension, 863
setdata5di16
mo_netcdf, 566
mo_netcdf::ncdimension, 863
setdata5di32
mo_netcdf, 566
mo_netcdf::ncdimension, 863
setdata5di64
mo_netcdf, 567
mo_netcdf::ncdimension, 863
setdata5di8
mo_netcdf, 567
mo_netcdf::ncdimension, 863
setdatascalarf32
mo_netcdf, 568
mo_netcdf::ncdimension, 863
setdatascalarf64
mo_netcdf, 568
mo_netcdf::ncdimension, 863
setdatascalari16
mo_netcdf, 569
mo_netcdf::ncdimension, 863
setdatascalari32
mo_netcdf, 569
mo_netcdf::ncdimension, 864
setdatascalari64
mo_netcdf, 569
mo_netcdf::ncdimension, 864
setdatascalari8
mo_netcdf, 570
mo_netcdf::ncdimension, 864
setdimension
mo_netcdf, 570
mo_netcdf::ncdataset, 844
setfillvalue
mo_netcdf::ncdimension, 864
setglobalattributechar
mo_netcdf, 571
mo_netcdf::ncdataset, 844
setglobalattributef32
mo_netcdf, 571
mo_netcdf::ncdataset, 844
setglobalattributef64
mo_netcdf, 571
mo_netcdf::ncdataset, 844
setglobalattributei16
mo_netcdf, 572
mo_netcdf::ncdataset, 845
setglobalattributei32
mo_netcdf, 572
mo_netcdf::ncdataset, 845
setglobalattributei64
mo_netcdf, 573
mo_netcdf::ncdataset, 845
setglobalattributei8
mo_netcdf, 573
mo_netcdf::ncdataset, 845
setup_description
mo_common_variables, 129
setvariable
mo_netcdf::ncdataset, 845
setvariableattributechar
mo_netcdf, 573
mo_netcdf::ncdimension, 864
setvariableattributef32
mo_netcdf, 574
mo_netcdf::ncdimension, 864
setvariableattributef64
mo_netcdf, 574
mo_netcdf::ncdimension, 865
setvariableattributei16
mo_netcdf, 575
mo_netcdf::ncdimension, 865
setvariableattributei32
mo_netcdf, 575

mo_netcdf::ncdimension, 865
setvariableattributei64
mo_netcdf, 575
mo_netcdf::ncdimension, 865
setvariableattributei8
mo_netcdf, 576
mo_netcdf::ncdimension, 865
setvariablefillvaluef32
mo_netcdf, 576
mo_netcdf::ncdimension, 865
setvariablefillvaluef64
mo_netcdf, 576
mo_netcdf::ncdimension, 865
setvariablefillvaluei16
mo_netcdf, 576
mo_netcdf::ncdimension, 865
setvariablefillvaluei32
mo_netcdf, 577
mo_netcdf::ncdimension, 865
setvariablefillvaluei64
mo_netcdf, 577
mo_netcdf::ncdimension, 866
setvariablefillvaluei8
mo_netcdf, 577
mo_netcdf::ncdimension, 866
setvariablewithids
mo_netcdf, 577
mo_netcdf::ncdataset, 846
setvariablewithnames
mo_netcdf, 578
mo_netcdf::ncdataset, 846
setvariablewithtypes
mo_netcdf, 579
mo_netcdf::ncdataset, 846
sigma_dp
mo_constants, 137
sigma_sp
mo_constants, 137
simper
mo_common_mhm_mrm_variables, 113
simulation_type
mo_common_variables, 129
single_objective_runoff
mo_mrm_objective_function_runoff, 399
skewness_dp
mo_moment, 269
mo_moment::skewness, 913
skewness_sp
mo_moment, 270
mo_moment::skewness, 913
slope_satpressure
mo_pet, 639
snow_acc_melt_param
mo_multi_param_reg, 466
snow_accum_melt
mo_snow_accum_melt, 684
soil_moisture
mo_soil_moisture, 690
soildb
mo_mpr_global_variables, 294
solarconst_dp
mo_constants, 137
solarconst_sp
mo_constants, 137
sort_index_dp
mo_orderpack, 629
mo_orderpack::sort_index, 919
sort_index_i4
mo_orderpack, 629
mo_orderpack::sort_index, 920
sort_index_sp
mo_orderpack, 630
mo_orderpack::sort_index, 920
sort_matrix
mo_sce, 682
sp
mo_kind, 231
sp2str
mo_string_utils, 705
mo_string_utils::num2str, 873
spatial_aggregation_3d
mo_spatial_agg_disagg_forcing, 693
mo_spatial_agg_disagg_forcing::spatial_aggregation, 920
spatial_aggregation_4d
mo_spatial_agg_disagg_forcing, 693
mo_spatial_agg_disagg_forcing::spatial_aggregation, 921
spatial_disaggregation_3d
mo_spatial_agg_disagg_forcing, 693
mo_spatial_agg_disagg_forcing::spatial_disaggregation, 922
spatial_disaggregation_4d
mo_spatial_agg_disagg_forcing, 694
mo_spatial_agg_disagg_forcing::spatial_disaggregation, 922
spc
mo_kind, 232
specheatet_dp
mo_constants, 137
specheatet_sp
mo_constants, 137
special_value_dp
mo_utils, 732
mo_utils::special_value, 923
special_value_sp
mo_utils, 733
mo_utils::special_value, 923
splitstring
mo_string_utils, 705
sqrt2
mo_constants, 137
sqrt2_d
mo_constants, 138
sqrt2_dp
mo_constants, 138

sqrt2_sp
mo_constants, 138
sse_dp_1d
mo_errormeasures, 170
mo_errormeasures::sse, 927
sse_dp_2d
mo_errormeasures, 170
mo_errormeasures::sse, 927
sse_dp_3d
mo_errormeasures, 171
mo_errormeasures::sse, 927
sse_sp_1d
mo_errormeasures, 172
mo_errormeasures::sse, 927
sse_sp_2d
mo_errormeasures, 172
mo_errormeasures::sse, 927
sse_sp_3d
mo_errormeasures, 173
mo_errormeasures::sse, 927
standard_score_dp
mo_standard_score, 696
mo_standard_score::standard_score, 929
standard_score_sp
mo_standard_score, 697
mo_standard_score::standard_score, 929
start
mo_ncwrite::variable, 952
startswith
mo_string_utils, 705
status
mo_timer, 722
stboltzmann
mo_mhm_constants, 258
stddev_dp
mo_moment, 270
mo_moment::stddev, 929
stddev_sp
mo_moment, 270
mo_moment::stddev, 929
steps
mo_mrm_write_fluxes_states::outputdataset, 878
mo_write_fluxes_states::outputdataset, 882
store
mo_mrm_write_fluxes_states::outputdataset, 878
mo_mrm_write_fluxes_states::outputvariable, 887
mo_write_fluxes_states::outputdataset, 882
mo_write_fluxes_states::outputvariable, 892
str2num
mo_string_utils, 706
sum_counter
mo_mrm_river_head, 419
swap_1d_dpc
mo_corr, 145
mo_corr::swap, 930
swap_1d_spc
mo_corr, 145
mo_corr::swap, 930
swap_vec_dp
mo_utils, 733
mo_utils::swap, 931
swap_vec_i4
mo_utils, 733
mo_utils::swap, 931
swap_vec_sp
mo_utils, 734
mo_utils::swap, 931
swap_xy_dp
mo_utils, 734
mo_utils::swap, 931
swap_xy_i4
mo_utils, 734
mo_utils::swap, 931
swap_xy_sp
mo_utils, 734
mo_utils::swap, 932

t0_dp
mo_constants, 138
t0_sp
mo_constants, 138
tabularintegralafast
mo_neutrons, 587
temporal_disagg_forcing
mo_temporal_disagg_forcing, 712
tetens_c1
mo_mhm_constants, 258
tetens_c2
mo_mhm_constants, 258
tetens_c3
mo_mhm_constants, 258
the
mo_mrm_write_fluxes_states::outputvariable, 887
mo_netcdf::ncdataset, 847
mo_netcdf::ncdimension, 866, 867
mo_write_fluxes_states::outputvariable, 892
thetafc
mo_mpr_global_variables::soiltype, 915
thetafc_till
mo_mpr_global_variables::soiltype, 915
thetapw
mo_mpr_global_variables::soiltype, 915
thetapw_till
mo_mpr_global_variables::soiltype, 916
thetas
mo_mpr_global_variables::soiltype, 916
thetas_till
mo_mpr_global_variables::soiltype, 916
tillagedepth
mo_mpr_global_variables, 294
time
mo_mrm_write_fluxes_states::outputdataset, 878
mo_write_fluxes_states::outputdataset, 882
time_counter
mo_mrm_river_head, 419
timer_check
mo_timer, 714

timer_clear
mo_timer, 715
timer_get
mo_timer, 716
timer_print
mo_timer, 717
timer_start
mo_timer, 718
timer_stop
mo_timer, 719
timers_init
mo_timer, 720
timestep
mo_common_mhm_mrm_variables, 113
timestep_et_input
mo_global_variables, 191
timestep_lai_input
mo_mpr_global_variables, 294
timestep_model_inputs
mo_global_variables, 191
timestep_model_outputs
mo_global_variables, 191
timestep_model_outputs_mrm
mo_mrm_global_variables, 345
timestep_neutrons_input
mo_global_variables, 191
timestep_sm_input
mo_global_variables, 191
to
mo_mrm_write_fluxes_states::outputdataset, 879
mo_mrm_write_fluxes_states::outputvariable, 887
mo_write_fluxes_states::outputdataset, 883
mo_write_fluxes_states::outputvariable, 892
tolower
mo_string_utils, 706
toupper
mo_string_utils, 707
twopi
mo_constants, 138
twopi_d
mo_constants, 138
twopi_dp
mo_constants, 138
twopi_sp
mo_constants, 139
twothird_dp
mo_constants, 139
twothird_sp
mo_constants, 139
tws
mo_global_variables::twsstructure, 933

uaspect
mo_mpr_file, 283
uconfig
mo_common_file, 101
mo_mrm_file, 331
ud
mo_mpr_global_variables::soiltype, 916
udaily_discharge
mo_mrm_file, 331
udefoutput
mo_file, 176
mo_mrm_file, 331
udem
mo_common_file, 101
udischarge
mo_mrm_file, 331
ufacc
mo_mrm_file, 331
ufdir
mo_mrm_file, 332
ugaugeloc
mo_mrm_file, 332
ugeolut
mo_mpr_file, 283
uhydrogeoclass
mo_mpr_file, 283
ulaiclass
mo_mpr_file, 284
ulailut
mo_mpr_file, 284
ulcoverclass
mo_common_file, 102
umeteo
mo_mpr_file, 284
umeteo_header
mo_mpr_file, 284
unamelist_mhm
mo_file, 176
unamelist_mhm_param
mo_file, 176
unamelist_mpr
mo_mpr_file, 284
unamelist_mpr_param
mo_mpr_file, 284
unamelist_mrm
mo_mrm_file, 332
unamelist_param_mrm
mo_mrm_file, 332
unlimited
mo_ncwrite::variable, 952
unpack_field_and_write_1d_dp
mo_mpr_restart, 303
mo_mpr_restart::unpack_field_and_write, 938
mo_restart, 669
mo_restart::unpack_field_and_write, 936
unpack_field_and_write_1d_i4
mo_mpr_restart, 303
mo_mpr_restart::unpack_field_and_write, 938
mo_restart, 669
mo_restart::unpack_field_and_write, 937
unpack_field_and_write_2d_dp
mo_mpr_restart, 303
mo_mpr_restart::unpack_field_and_write, 939
mo_restart, 670
mo_restart::unpack_field_and_write, 937

unpack_field_and_write_3d_dp
mo_mpr_restart, 304
mo_mpr_restart::unpack_field_and_write, 939
mo_restart, 670
mo_restart::unpack_field_and_write, 937
uopti
mo_common_mhm_mrm_file, 104
uopti_nml
mo_common_mhm_mrm_file, 104
updatedataset
mo_mrm_write_fluxes_states, 451
mo_mrm_write_fluxes_states::outputdataset, 877
mo_write_fluxes_states, 743
mo_write_fluxes_states::outputdataset, 881
updatevariable
mo_mrm_write_fluxes_states, 452
mo_mrm_write_fluxes_states::outputvariable, 885, 888
mo_write_fluxes_states, 744
mo_write_fluxes_states::outputvariable, 890, 893
upper_bound
mo_common_variables::gridremapper, 801
upscale_arithmetic_mean
mo_upscaling_operators, 724
upscale_geometric_mean
mo_upscaling_operators, 726
upscale_harmonic_mean
mo_upscaling_operators, 727
upscale_p_norm
mo_upscaling_operators, 728
uslope
mo_mpr_file, 285
mo_mrm_file, 332
usoil_database
mo_mpr_file, 285
usoilclass
mo_mpr_file, 285
utws
mo_file, 176

v
mo_ncwrite, 514
val
mo_kind::sprs2_dp, 925
mo_kind::sprs2_sp, 926
values
mo_ncwrite::attribute, 763
var2nc_1d_dp
mo_ncwrite, 502
mo_ncwrite::var2nc, 942
var2nc_1d_i4
mo_ncwrite, 503
mo_ncwrite::var2nc, 943
var2nc_1d_sp
mo_ncwrite, 504
mo_ncwrite::var2nc, 943
var2nc_2d_dp
mo_ncwrite, 504
mo_ncwrite::var2nc, 943
var2nc_2d_i4
mo_ncwrite, 505
mo_ncwrite::var2nc, 943
var2nc_2d_sp
mo_ncwrite, 506
mo_ncwrite::var2nc, 944
var2nc_3d_dp
mo_ncwrite, 506
mo_ncwrite::var2nc, 944
var2nc_3d_i4
mo_ncwrite, 507
mo_ncwrite::var2nc, 944
var2nc_3d_sp
mo_ncwrite, 508
mo_ncwrite::var2nc, 945
var2nc_4d_dp
mo_ncwrite, 508
mo_ncwrite::var2nc, 945
var2nc_4d_i4
mo_ncwrite, 509
mo_ncwrite::var2nc, 945
var2nc_4d_sp
mo_ncwrite, 510
mo_ncwrite::var2nc, 946
var2nc_5d_dp
mo_ncwrite, 510
mo_ncwrite::var2nc, 946
var2nc_5d_i4
mo_ncwrite, 511
mo_ncwrite::var2nc, 946
var2nc_5d_sp
mo_ncwrite, 512
mo_ncwrite::var2nc, 947
variable
mo_mrm_write_fluxes_states::outputvariable, 888
mo_write_fluxes_states::outputvariable, 893
variables
mo_mrm_write_fluxes_states::outputdataset, 879
mo_write_fluxes_states::outputdataset, 883
variables_alloc
mo_init_states, 198
variables_alloc_routing
mo_mrm_init, 352
variables_default_init
mo_init_states, 200
variables_default_init_routing
mo_mrm_init, 353
variance_dp
mo_moment, 270
mo_moment::variance, 953
variance_sp
mo_moment, 270
mo_moment::variance, 953
varid
mo_ncwrite::variable, 953
varnametotalrunoff
mo_mrm_global_variables, 345
vars

mo_mrm_write_fluxes_states::outputdataset, 879
mo_write_fluxes_states::outputdataset, 883
version
mo_file, 177
mo_mpr_file, 285
mo_mrm_file, 332
version_date
mo_file, 177
mo_mpr_file, 285
mo_mrm_file, 332
vgenuchten_sandtresh
mo_mpr_constants, 274
vgenuchtenn_c1
mo_mpr_constants, 274
vgenuchtenn_c10
mo_mpr_constants, 274
vgenuchtenn_c11
mo_mpr_constants, 274
vgenuchtenn_c12
mo_mpr_constants, 275
vgenuchtenn_c13
mo_mpr_constants, 275
vgenuchtenn_c14
mo_mpr_constants, 275
vgenuchtenn_c15
mo_mpr_constants, 275
vgenuchtenn_c16
mo_mpr_constants, 275
vgenuchtenn_c17
mo_mpr_constants, 275
vgenuchtenn_c18
mo_mpr_constants, 275
vgenuchtenn_c2
mo_mpr_constants, 275
vgenuchtenn_c3
mo_mpr_constants, 276
vgenuchtenn_c4
mo_mpr_constants, 276
vgenuchtenn_c5
mo_mpr_constants, 276
vgenuchtenn_c6
mo_mpr_constants, 276
vgenuchtenn_c7
mo_mpr_constants, 276
vgenuchtenn_c8
mo_mpr_constants, 276
vgenuchtenn_c9
mo_mpr_constants, 276

warmingdays
mo_common_mhm_mrm_variables, 113
warmper
mo_common_mhm_mrm_variables, 114
wd
mo_mpr_global_variables::soiltype, 916
wflag
mo_ncwrite::variable, 953
which
mo_mrm_write_fluxes_states::outputvariable, 888
mo_write_fluxes_states::outputvariable, 893
windmeasheight
mo_mpr_constants, 277
wnse_dp_1d
mo_errormeasures, 174
mo_errormeasures::wnse, 954
wnse_dp_2d
mo_errormeasures, 174
mo_errormeasures::wnse, 954
wnse_dp_3d
mo_errormeasures, 174
mo_errormeasures::wnse, 954
wnse_sp_1d
mo_errormeasures, 174
mo_errormeasures::wnse, 954
wnse_sp_2d
mo_errormeasures, 174
mo_errormeasures::wnse, 954
wnse_sp_3d
mo_errormeasures, 174
mo_errormeasures::wnse, 955
write
mo_mrm_write_fluxes_states::outputdataset, 879
mo_write_fluxes_states::outputdataset, 883
write_best_final
mo_sce.f90, 1018
write_best_intermediate
mo_sce.f90, 1019
write_configfile
mo_mrm_write, 443
mo_write_ascii, 735
write_daily_obs_sim_discharge
mo_mrm_write, 444
write_dynamic_netcdf
mo_ncwrite, 512
write_eff_params
mo_mpr_restart, 304
write_grid_info
mo_common_restart, 121
write_mpr_restart_files
mo_mpr_restart, 305
write_optifile
mo_write_ascii, 736
write_optinamelist
mo_write_ascii, 737
write_population
mo_sce.f90, 1019
write_restart
mo_common_variables, 129
write_restart_files
mo_restart, 670
write_static_netcdf
mo_ncwrite, 513
write_termination_case
mo_sce.f90, 1019
writes
mo_mrm_write_fluxes_states::outputvariable, 888
mo_write_fluxes_states::outputvariable, 893

writetimestep
mo_mrm_write_fluxes_states, 453
mo_mrm_write_fluxes_states::outputdataset, 877
mo_write_fluxes_states, 745
mo_write_fluxes_states::outputdataset, 881
writevariableattributes
mo_mrm_write_fluxes_states, 453
mo_write_fluxes_states, 746
writevariabletimestep
mo_mrm_write_fluxes_states, 454
mo_mrm_write_fluxes_states::outputvariable, 885
mo_write_fluxes_states, 746
mo_write_fluxes_states::outputvariable, 890
writing
mo_mrm_write_fluxes_states::outputvariable, 888
mo_write_fluxes_states::outputvariable, 893
written
mo_mrm_write_fluxes_states::outputdataset, 879
mo_write_fluxes_states::outputdataset, 883

x
mo_common_variables::grid, 799
xllcorner
mo_common_variables::grid, 799
xor4096d_0d
mo_xor4096, 748
mo_xor4096::xor4096, 955
xor4096d_1d
mo_xor4096, 748
mo_xor4096::xor4096, 955
xor4096f_0d
mo_xor4096, 749
mo_xor4096::xor4096, 955
xor4096f_1d
mo_xor4096, 749
mo_xor4096::xor4096, 956
xor4096gd_0d
mo_xor4096, 749
mo_xor4096::xor4096g, 957
xor4096gd_1d
mo_xor4096, 749
mo_xor4096::xor4096g, 957
xor4096gf_0d
mo_xor4096, 749
mo_xor4096::xor4096g, 957
xor4096gf_1d
mo_xor4096, 750
mo_xor4096::xor4096g, 957
xor4096l_0d
mo_xor4096, 750
mo_xor4096::xor4096, 956
xor4096l_1d
mo_xor4096, 750
mo_xor4096::xor4096, 956
xor4096s_0d
mo_xor4096, 750
mo_xor4096::xor4096, 956
xor4096s_1d
mo_xor4096, 750
mo_xor4096::xor4096, 956
xtype
mo_ncwrite::attribute, 763
mo_ncwrite::variable, 953

y
mo_common_variables::grid, 799
year_counter
mo_mrm_write, 446
yeardays
mo_common_constants, 100
mo_constants, 139
yearmonths
mo_common_constants, 100
mo_constants, 139
yearmonths_i4
mo_common_constants, 100
yend
mo_common_variables::period, 901
yllcorner
mo_common_variables::grid, 799
ystart
mo_common_variables::period, 902

zeroflowratio
mo_mrm_signatures, 435
zroots_unity_dp
mo_corr, 145
zroots_unity_sp
mo_corr, 145