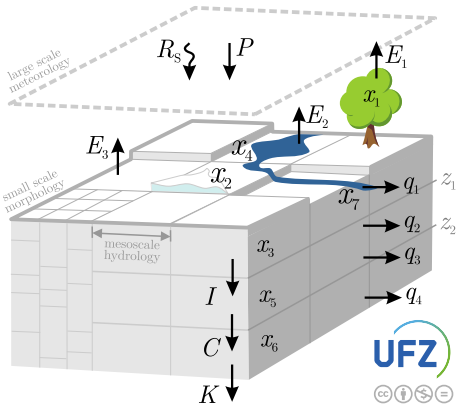
HELMHOLTZ
Centre for Environmental Research

The mesoscale Hydrologic Model

mHM



Documentation for version 5.11.2

Presented by the mHM Community

Project Leader: Luis Samaniego

Supervisor: Sabine Attinger

Authors & Contributors
(arranged in alphabetical order)

Johannes Brenner, John Craven, Matthias Cuntz, Giovanni Dalmasso, Cüneyd M. Demirel, Nicola Döring, Miao Jing, Maren Kaluza, Rohini Kumar, Ben Langenberg, Juliane Mai, Sebastian Müller, Jude Musuuza, Vladyslav Prykhodko, Oldrich Rakovec, David Schäfer, Christoph Schneider, Martin Schrön, Lennart Schüler, Robert Schweppe, Pallav Shrestha, Diana Spieler, Simon Stisen, Stephan Thober, 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
2.3 Run mHM under CYGWIN on Windows . . . 9
2.4 Compile mHM . . . 13
2.5 Test mHM on Example Basin . . . 13
2.6 Run your own Simulation . . . 13
2.7 Calibration and Optimization . . . 14

3 Data Preparation for mHM **17**

3.1 Getting Started . . . 17
3.2 Preparation of the Forcings . . . 19
3.3 Preparation of the LatLon Grid . . . 20
3.4 Preparation of the Morphological Data . . . 21
3.5 A possible GIS workflow . . . 21
3.6 Land Cover Data . . . 29
3.7 Table Data . . . 30
3.8 Post-GIS preparation . . . 32

4 Visualizing Model Output **33**

5 Calibration Options **35**

5.1 Calibration of Parameters on Observations . . . 35

6 Coding and Documentation Style 37

7 The details about the test basin 41

8 Protocols for setting up a new mHM basin 43

8.1 Check for possible input data errors using mHM outputs . . . 43

8.2 Protocol for generating a restart file . . . 44

8.3 Protocol for determining the warm-up period for a new basin . . . 44

9 mRM - the Multiscale Routing Model: Running the stand-alone version of the routing model 45

9.1 Introduction to mRM . . . 45

9.2 Getting Started . . . 47

9.3 Data preparation for mRM . . . 49

10 mHM Dependencies 51

10.1 NetCDF4 . . . 51

11 Install Instruction 53

11.1 Dependencies . . . 53

11.2 System-dependent installation instructions . . . 53

11.3 Specific setups . . . 56

11.4 Installation . . . 57

11.5 Building Realease or Debug versions . . . 58

11.6 Troubleshooting . . . 58

12 Publications using mHM 59

13 mHM Release Notes 63

13.1 mHM v5.11.2 (Jul 2021) . . . 63

13.2 mHM v5.11.1 (Mar 2021) . . . 63

13.3 mHM v5.11 (Feb 2021) . . . 64

13.4 mHM v5.10 (June 2019) . . . 65

13.5 mHM v5.9 (July 2018) . . . 66

13.6 mHM v5.8 (Dec 2017) . . . 68

13.7 mHM v5.7 (Jun 2017) . . . 68

13.8 mHM v5.6 (Dec 2016) . . . 69

13.9 mHM v5.5 (Jun 2016) . . . 70

13.10 mHM v5.4 (Dec 2015) . . . 70

13.11 mHM v5.3 (Jun 2015) . . . 71

13.12 mHM 5.2 (Dec 2014) . . . 72

13.13 mHM 5.1 (Jun 2014) . . . 73

13.14 mHM 5.0 (Dec 2013) . . . 73

14 Modules Index **75**

14.1 Modules List . . . 75

15 Data Type Index **79**

15.1 Data Types List . . . 79

16 File Index **81**

16.1 File List . . . 81

17 Module Documentation **83**

17.1 dummy_mpr Module Reference . . . 83
17.2 mo_canopy_interc Module Reference . . . 83
17.3 mo_check Module Reference . . . 84
17.4 mo_common_constants Module Reference . . . 85
17.5 mo_common_datetime_type Module Reference . . . 89
17.6 mo_common_file Module Reference . . . 90
17.7 mo_common_functions Module Reference . . . 91
17.8 mo_common_mhm_mrm_file Module Reference . . . 92
17.9 mo_common_mhm_mrm_mpi_tools Module Reference . . . 94
17.10 mo_common_mhm_mrm_read_config Module Reference . . . 95
17.11 mo_common_mhm_mrm_restart Module Reference . . . 99
17.12 mo_common_mhm_mrm_variables Module Reference . . . 100
17.13 mo_common_read_config Module Reference . . . 107
17.14 mo_common_read_data Module Reference . . . 111
17.15 mo_common_restart Module Reference . . . 113
17.16 mo_common_variables Module Reference . . . 116
17.17 mo_file Module Reference . . . 124
17.18 mo_global_variables Module Reference . . . 127
17.19 mo_grid Module Reference . . . 141
17.20 mo_init_states Module Reference . . . 148
17.21 mo_meteo_forcings Module Reference . . . 150
17.22 mo_mhm Module Reference . . . 157
17.23 mo_mhm_constants Module Reference . . . 163
17.24 mo_mhm_eval Module Reference . . . 168
17.25 mo_mhm_read_config Module Reference . . . 171
17.26 mo_mpr_constants Module Reference . . . 173
17.27 mo_mpr_eval Module Reference . . . 181
17.28 mo_mpr_file Module Reference . . . 183
17.29 mo_mpr_global_variables Module Reference . . . 189

17.30 mo_mpr_pet Module Reference . . . 200
17.31 mo_mpr_read_config Module Reference . . . 206
17.32 mo_mpr_restart Module Reference . . . 207
17.33 mo_mpr_runoff Module Reference . . . 211
17.34 mo_mpr_smhorizons Module Reference . . . 214
17.35 mo_mpr_soilmoist Module Reference . . . 216
17.36 mo_mpr_startup Module Reference . . . 223
17.37 mo_mrm_constants Module Reference . . . 228
17.38 mo_mrm_file Module Reference . . . 229
17.39 mo_mrm_global_variables Module Reference . . . 235
17.40 mo_mrm_init Module Reference . . . 250
17.41 mo_mrm_mpr Module Reference . . . 258
17.42 mo_mrm_net_startup Module Reference . . . 261
17.43 mo_mrm_objective_function_runoff Module Reference . . . 276
17.44 mo_mrm_pre_routing Module Reference . . . 306
17.45 mo_mrm_read_config Module Reference . . . 311
17.46 mo_mrm_read_data Module Reference . . . 314
17.47 mo_mrm_restart Module Reference . . . 319
17.48 mo_mrm_riv_temp_class Module Reference . . . 323
17.49 mo_mrm_river_head Module Reference . . . 333
17.50 mo_mrm_routing Module Reference . . . 338
17.51 mo_mrm_signatures Module Reference . . . 343
17.52 mo_mrm_write Module Reference . . . 352
17.53 mo_mrm_write_fluxes_states Module Reference . . . 358
17.54 mo_multi_param_reg Module Reference . . . 366
17.55 mo_neutrons Module Reference . . . 378
17.56 mo_objective_function Module Reference . . . 386
17.57 mo_optimization Module Reference . . . 409
17.58 mo_pet Module Reference . . . 410
17.59 mo_prepare_gridded_lai Module Reference . . . 417
17.60 mo_read_latlon Module Reference . . . 420
17.61 mo_read_lut Module Reference . . . 421
17.62 mo_read_nc Module Reference . . . 424
17.63 mo_read_optional_data Module Reference . . . 429
17.64 mo_read_spatial_data Module Reference . . . 431
17.65 mo_read_timeseries Module Reference . . . 434
17.66 mo_read_wrapper Module Reference . . . 435
17.67 mo_restart Module Reference . . . 438
17.68 mo_runoff Module Reference . . . 443
17.69 mo_snow_accum_melt Module Reference . . . 446

17.70 mo_soil_database Module Reference . . . 448
17.71 mo_soil_moisture Module Reference . . . 450
17.72 mo_spatial_agg_disagg_forcing Module Reference . . . 454
17.73 mo_startup Module Reference . . . 456
17.74 mo_template Module Reference . . . 460
17.75 mo_temporal_disagg_forcing Module Reference . . . 462
17.76 mo_upscaling_operators Module Reference . . . 467
17.77 mo_write_ascii Module Reference . . . 474
17.78 mo_write_fluxes_states Module Reference . . . 477

18 Data Type Documentation **487**

18.1 mo_common_mhm_mrm_restart::check_consistency_element Interface Reference . . . 487
18.2 mo_common_datetime_type::datetimeinfo Type Reference . . . 488
18.3 mo_common_variables::domain_meta Type Reference . . . 491
18.4 mo_mrm_global_variables::domaininfo_mrm Type Reference . . . 493
18.5 mo_mrm_global_variables::gaugingstation Type Reference . . . 495
18.6 mo_common_variables::grid Type Reference . . . 496
18.7 mo_common_variables::gridremapper Type Reference . . . 499
18.8 mo_template::mean Interface Reference . . . 500
18.9 mo_mrm_write_fluxes_states::outputdataset Interface Reference . . . 502
18.10 mo_write_fluxes_states::outputdataset Interface Reference . . . 505
18.11 mo_mrm_write_fluxes_states::outputvariable Interface Reference . . . 509
18.12 mo_write_fluxes_states::outputvariable Interface Reference . . . 514
18.13 mo_common_variables::period Type Reference . . . 518
18.14 mo_read_spatial_data::read_spatial_data_ascii Interface Reference . . . 520
18.15 mo_mrm_riv_temp_class::riv_temp_type Module Reference . . . 523
18.16 mo_mpr_global_variables::soiltype Type Reference . . . 538
18.17 mo_spatial_agg_disagg_forcing::spatial_aggregation Interface Reference . . . 541
18.18 mo_spatial_agg_disagg_forcing::spatial_disaggregation Interface Reference . . . 542
18.19 mo_mpr_restart::unpack_field_and_write Interface Reference . . . 543
18.20 mo_restart::unpack_field_and_write Interface Reference . . . 545

19 File Documentation **549**

19.1 doc/1-main.dox File Reference . . . 549
19.2 doc/2-get_started.dox File Reference . . . 549
19.3 doc/3-data_preparation.dox File Reference . . . 549
19.4 doc/4-visualise_out.dox File Reference . . . 549
19.5 doc/5-calibration.dox File Reference . . . 549
19.6 doc/6-style_guide.dox File Reference . . . 549
19.7 doc/7-test_basin.dox File Reference . . . 549
19.8 doc/8-protocols_for_setup_new_mHM_basin.dox File Reference . . . 549

19.9 doc/9-mRM.dox File Reference . . . 549
19.10 doc/DEPENDENCIES.md File Reference . . . 549
19.11 doc/INSTALL.md File Reference . . . 549
19.12 doc/mhm_papers.md File Reference . . . 549
19.13 doc/RELEASES.md File Reference . . . 549
19.14 src/common/mo_check.f90 File Reference . . . 549
19.15 src/common/mo_common_constants.f90 File Reference . . . 550
19.16 src/common/mo_common_datetime_type.f90 File Reference . . . 550
19.17 src/common/mo_common_file.f90 File Reference . . . 551
19.18 src/common/mo_common_functions.f90 File Reference . . . 551
19.19 src/common/mo_common_read_config.f90 File Reference . . . 551
19.20 src/common/mo_common_read_data.f90 File Reference . . . 552
19.21 src/common/mo_common_restart.f90 File Reference . . . 552
19.22 src/common/mo_common_variables.f90 File Reference . . . 552
19.23 src/common/mo_grid.f90 File Reference . . . 553
19.24 src/common/mo_read_latlon.f90 File Reference . . . 554
19.25 src/common/mo_read_nc.f90 File Reference . . . 554
19.26 src/common/mo_read_spatial_data.f90 File Reference . . . 555
19.27 src/common/mo_read_timeseries.f90 File Reference . . . 555
19.28 src/common/mo_template.f90 File Reference . . . 555
19.29 src/common_mHM_mRM/mo_common_mHM_mRM_file.f90 File Reference . . . 556
19.30 src/common_mHM_mRM/mo_common_mHM_mRM_MPI_tools.f90 File Reference . . . 556
19.31 src/common_mHM_mRM/mo_common_mHM_mRM_read_config.f90 File Reference . . . 557
19.32 src/common_mHM_mRM/mo_common_mHM_mRM_restart.f90 File Reference . . . 557
19.33 src/common_mHM_mRM/mo_common_mHM_mRM_variables.f90 File Reference . . . 557
19.34 src/common_mHM_mRM/mo_optimization.f90 File Reference . . . 558
19.35 src/mHM/mhm_driver.f90 File Reference . . . 558
19.36 src/mHM/mo_canopy_interc.f90 File Reference . . . 561
19.37 src/mHM/mo_file.f90 File Reference . . . 562
19.38 src/mHM/mo_global_variables.f90 File Reference . . . 562
19.39 src/mHM/mo_init_states.f90 File Reference . . . 564
19.40 src/mHM/mo_meteo_forcings.f90 File Reference . . . 564
19.41 src/mHM/mo_mhm.f90 File Reference . . . 564
19.42 src/mHM/mo_mhm_constants.f90 File Reference . . . 565
19.43 src/mHM/mo_mhm_eval.f90 File Reference . . . 566
19.44 src/mHM/mo_mhm_read_config.f90 File Reference . . . 566
19.45 src/mHM/mo_neutrons.f90 File Reference . . . 566
19.46 src/mHM/mo_objective_function.f90 File Reference . . . 567
19.47 src/mHM/mo_pet.f90 File Reference . . . 567
19.48 src/mHM/mo_read_optional_data.f90 File Reference . . . 568

19.49src/mHM/mo_restart.f90 File Reference . . . 568
19.50src/mHM/mo_runoff.f90 File Reference . . . 569
19.51src/mHM/mo_snow_accum_melt.f90 File Reference . . . 569
19.52src/mHM/mo_soil_moisture.f90 File Reference . . . 569
19.53src/mHM/mo_spatial_agg_disagg_forcing.f90 File Reference . . . 570
19.54src/mHM/mo_startup.f90 File Reference . . . 570
19.55src/mHM/mo_temporal_disagg_forcing.f90 File Reference . . . 570
19.56src/mHM/mo_write_ascii.f90 File Reference . . . 571
19.57src/mHM/mo_write_fluxes_states.f90 File Reference . . . 571
19.58src/MPR/mo_mpr_constants.f90 File Reference . . . 572
19.59src/MPR/mo_mpr_eval.f90 File Reference . . . 573
19.60src/MPR/mo_mpr_file.f90 File Reference . . . 573
19.61src/MPR/mo_mpr_global_variables.f90 File Reference . . . 574
19.62src/MPR/mo_mpr_pet.f90 File Reference . . . 576
19.63src/MPR/mo_mpr_read_config.f90 File Reference . . . 576
19.64src/MPR/mo_mpr_restart.f90 File Reference . . . 576
19.65src/MPR/mo_mpr_runoff.f90 File Reference . . . 577
19.66src/MPR/mo_mpr_smhorizons.f90 File Reference . . . 577
19.67src/MPR/mo_mpr_soilmoist.f90 File Reference . . . 577
19.68src/MPR/mo_mpr_startup.f90 File Reference . . . 578
19.69src/MPR/mo_multi_param_reg.f90 File Reference . . . 578
19.70src/MPR/mo_prepare_gridded_lai.f90 File Reference . . . 579
19.71src/MPR/mo_read_lut.f90 File Reference . . . 579
19.72src/MPR/mo_read_wrapper.f90 File Reference . . . 579
19.73src/MPR/mo_soil_database.f90 File Reference . . . 579
19.74src/MPR/mo_upscaling_operators.f90 File Reference . . . 580
19.75src/MPR/mpr_driver.f90 File Reference . . . 580
19.76src/mRM/mo_mrm_constants.f90 File Reference . . . 583
19.77src/mRM/mo_mrm_file.f90 File Reference . . . 583
19.78src/mRM/mo_mrm_global_variables.f90 File Reference . . . 584
19.79src/mRM/mo_mrm_init.f90 File Reference . . . 586
19.80src/mRM/mo_mrm_mpr.f90 File Reference . . . 586
19.81src/mRM/mo_mrm_net_startup.f90 File Reference . . . 586
19.82src/mRM/mo_mrm_objective_function_runoff.f90 File Reference . . . 588
19.83src/mRM/mo_mrm_pre_routing.f90 File Reference . . . 589
19.84src/mRM/mo_mrm_read_config.f90 File Reference . . . 589
19.85src/mRM/mo_mrm_read_data.f90 File Reference . . . 590
19.86src/mRM/mo_mrm_restart.f90 File Reference . . . 590
19.87src/mRM/mo_mrm_riv_temp_class.f90 File Reference . . . 590
19.88src/mRM/mo_mrm_river_head.f90 File Reference . . . 591

19.89src/mRM/mo_mrm_routing.f90 File Reference 592
19.90src/mRM/mo_mrm_signatures.f90 File Reference 592
19.91src/mRM/mo_mrm_write.f90 File Reference 593
19.92src/mRM/mo_mrm_write_fluxes_states.f90 File Reference 593

Bibliography **595**

Index **597**

Chapter 1

Introduction to mHM

Welcome to the documentation of mHM version 5.11.2

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] .

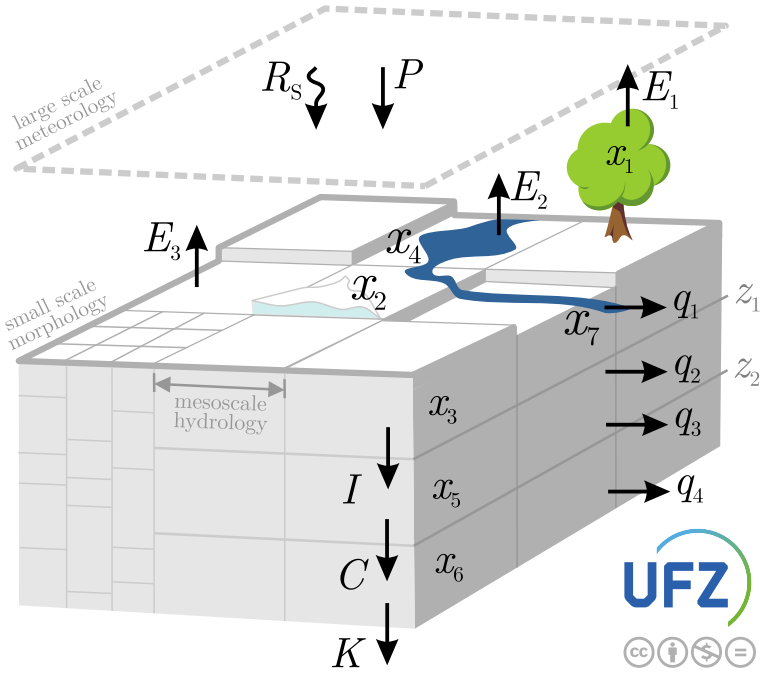


Figure 1.1: Typical mHM cell

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.

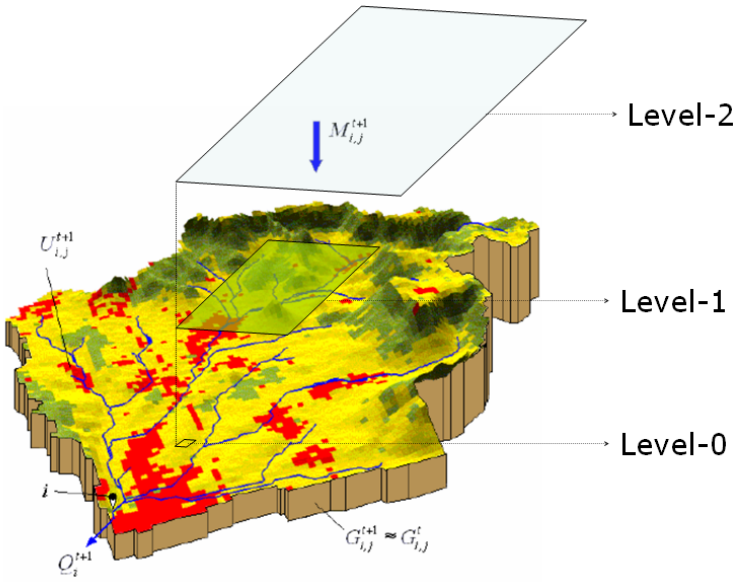


Figure 1.2: Hierarchy of data and modeling levels in mHM

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] .

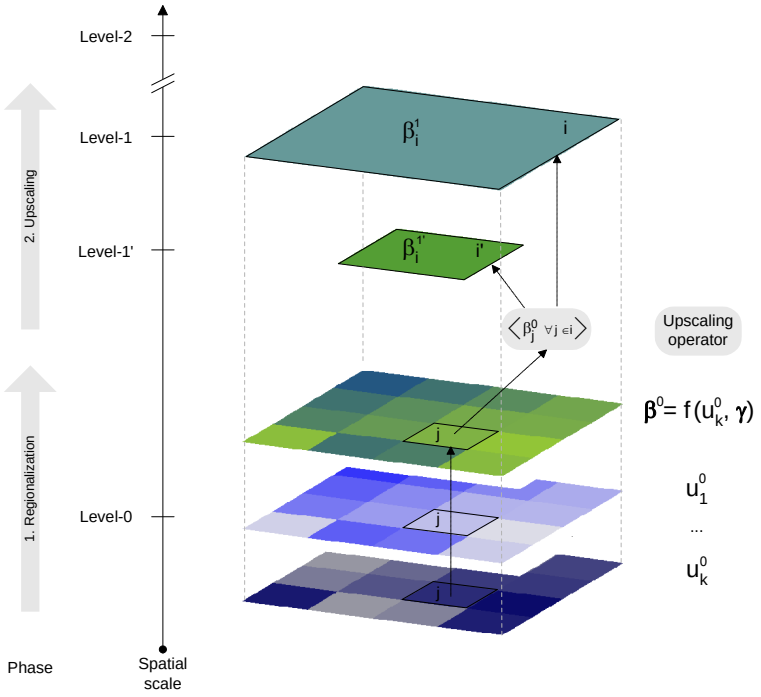


Figure 1.3: MPR

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

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

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:** When using ubuntu and gcc, you can use apt to install NETCDF:
  `sudo apt-get install libnetcdf-dev libnetcdff-dev`

- **on MacOS:** You can use Homebrew to install NETCDF but you need to specify the path in cmake later on:
  `brew install netcdf`

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/doc/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):

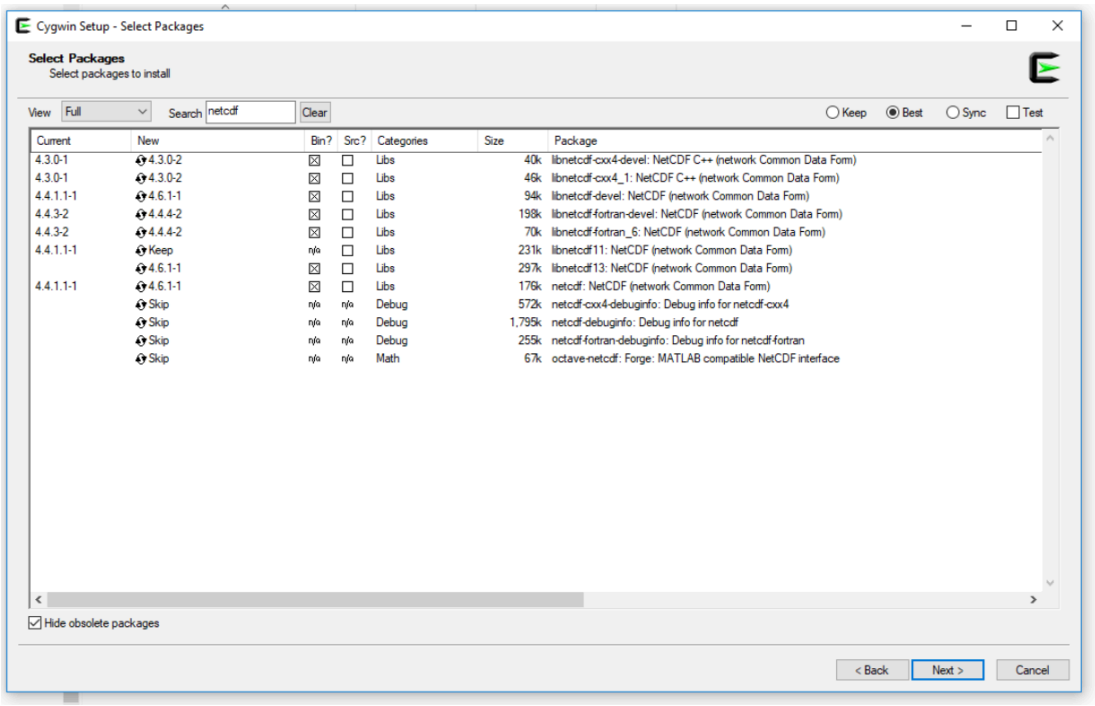


Figure 2.1: netcdf packages for CYGWIN

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

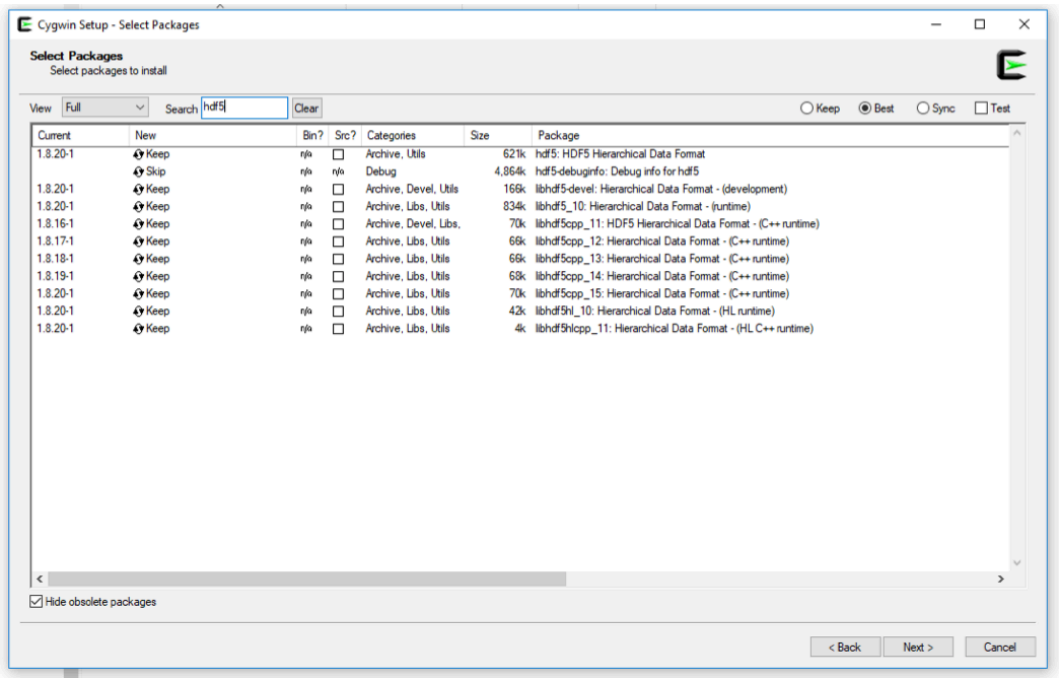


Figure 2.2: hdf5 packages for CYGWIN

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

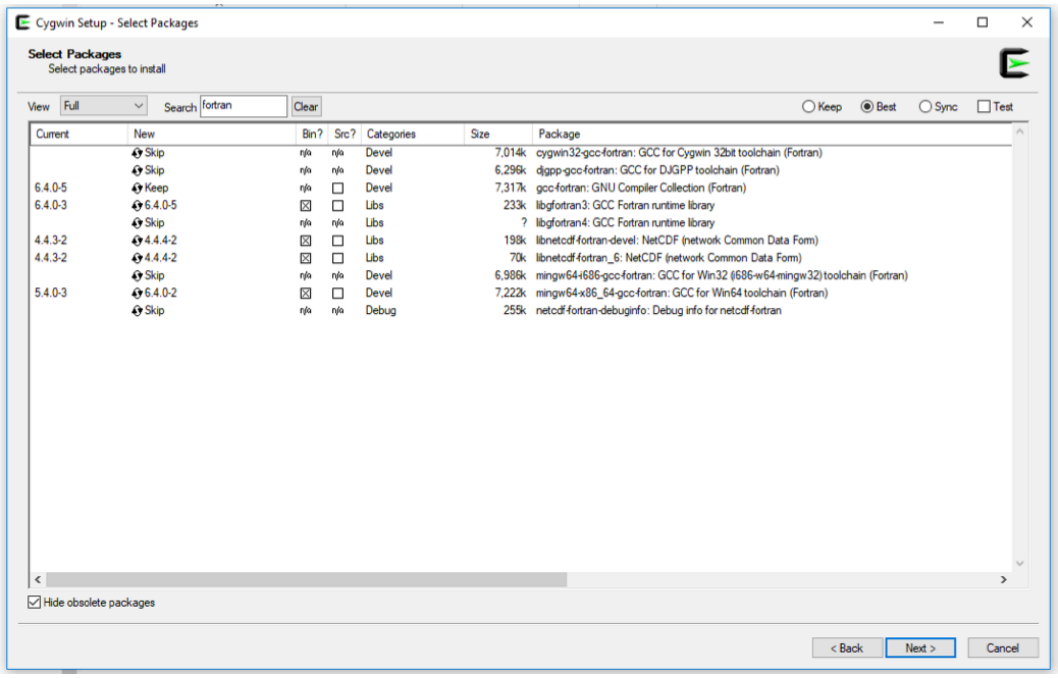


Figure 2.3: gfortran packages for CYGWIN

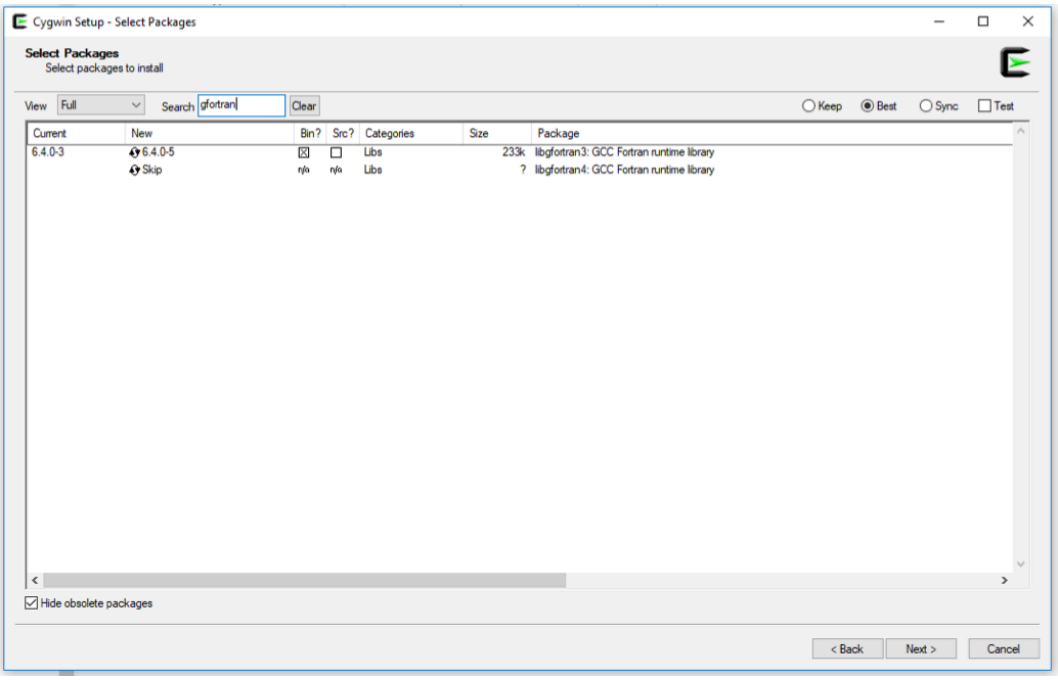


Figure 2.4: libgfortran packages for CYGWIN

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

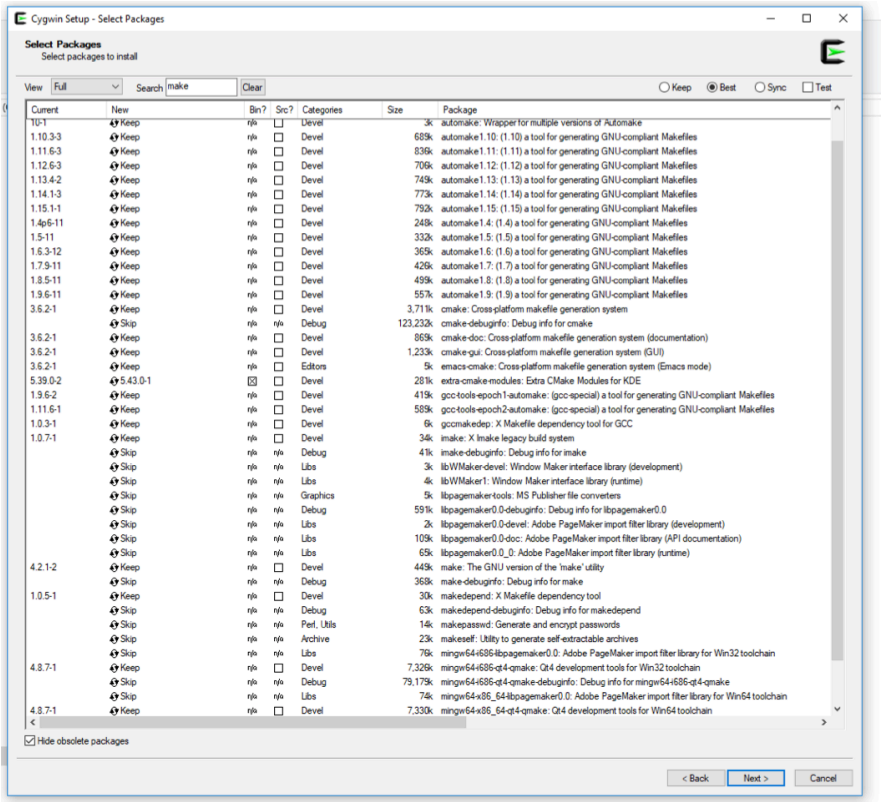


Figure 2.5: make packages for CYGWIN

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 `\cygwin64\bin` folder in your environmental path (fig_cygwin_path).

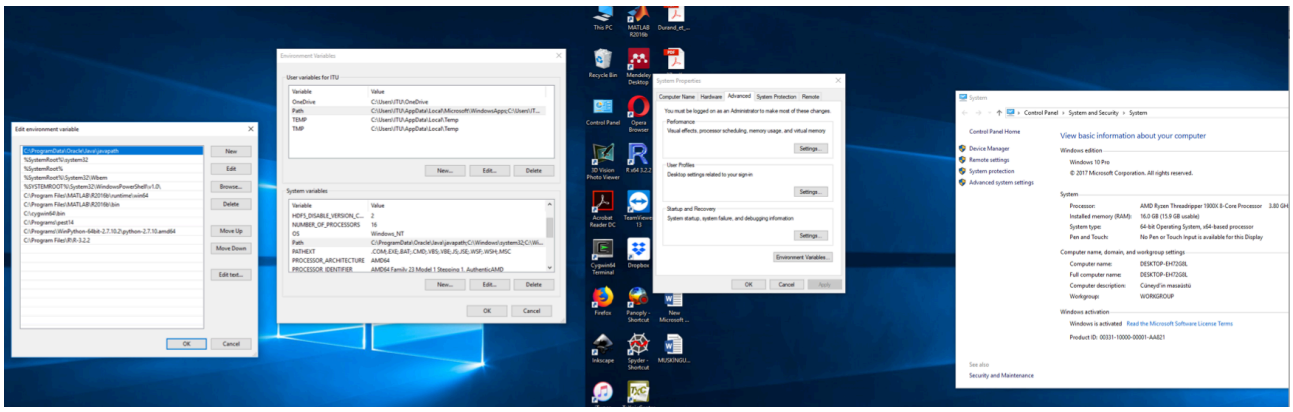


Figure 2.6: setup environmental path CYGWIN

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\bin` on the command line) under Windows. This will produce an executable `mhm\bin` which can be launched by `./mhm\bin` and results for the test basins can be found under `./test_basin/output_b1/` and `./test_basin_↩2/output_b1/`.

2.4 Compile mHM

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

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. This will also guarantee that the file ends with a new line character that is required by the fortran language standard. If you are writing these files with external programs, make sure that the written files comply with Fortran language standard, in particular make sure that the file ends with a new line.

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 \gamma, defined in `mhm_parameters.nml` . In order to fit discharge to observed data, mHM has to be recalibrated. This will optimise the \gamma 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 `FinalParams.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-WFDEI.↩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.

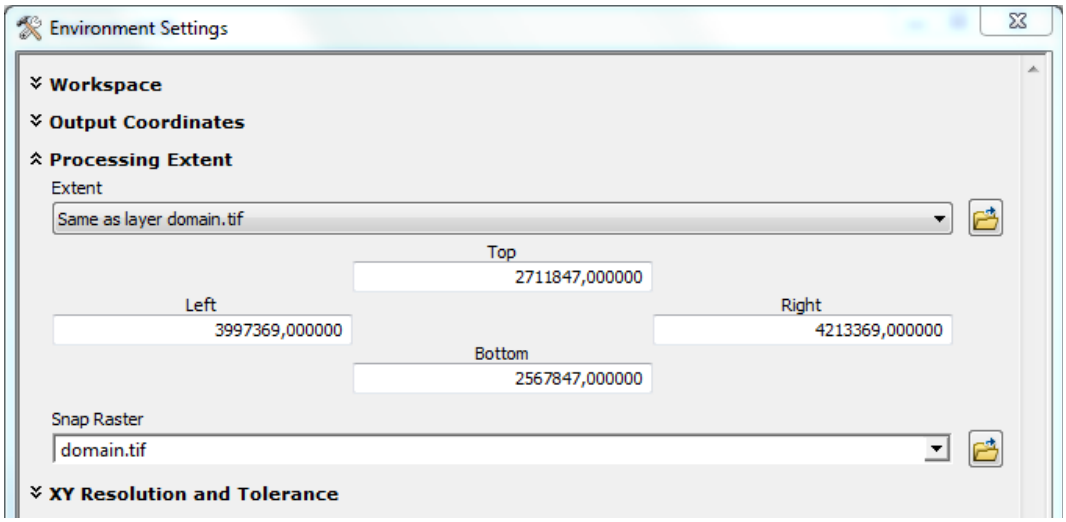


Figure 3.1: Button 'Environments...' in all Toolbox windows

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.

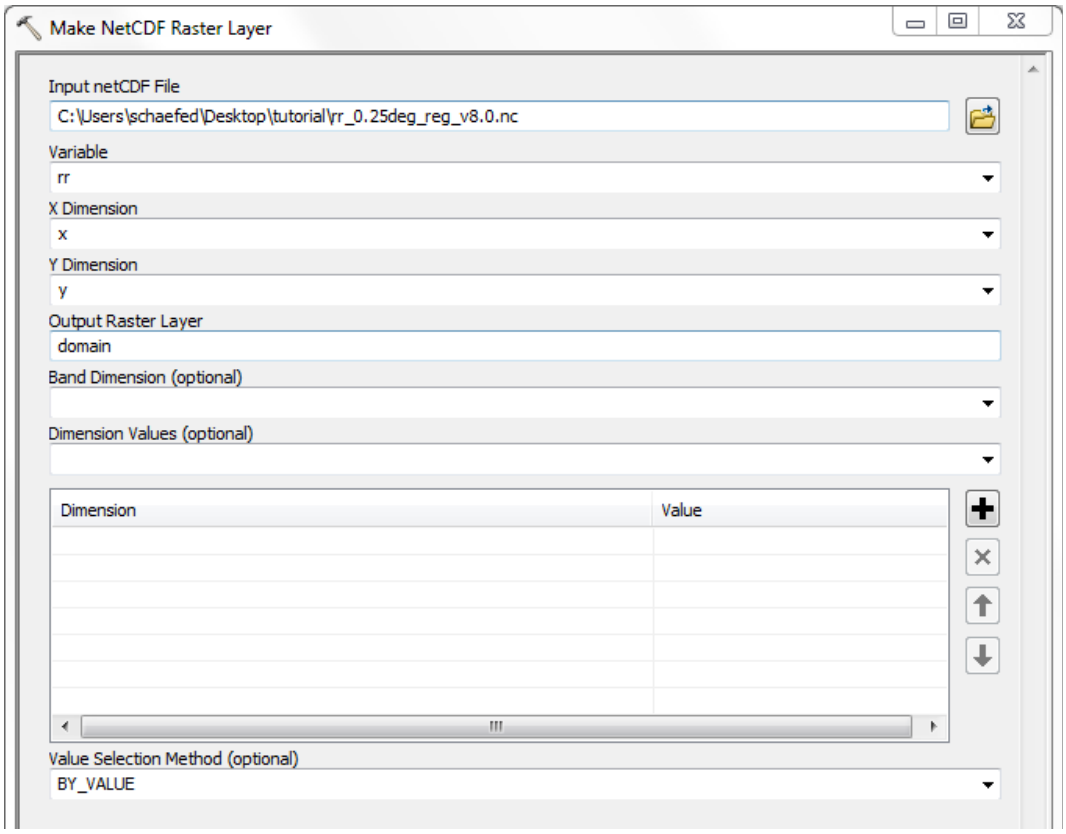


Figure 3.2: System Toolboxes -> Multidimension Tools -> Make NetCDF Raster Layer

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.

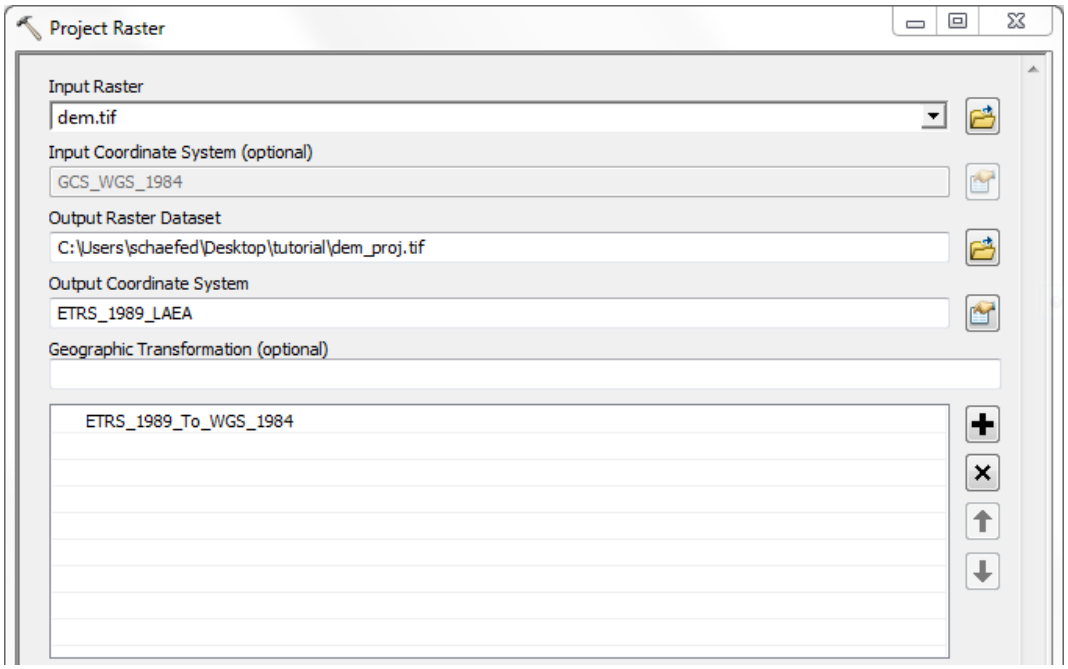


Figure 3.3: System Toolboxes -> Data Management Tools -> Projections and Transformations -> Raster -> Project Raster

- 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.

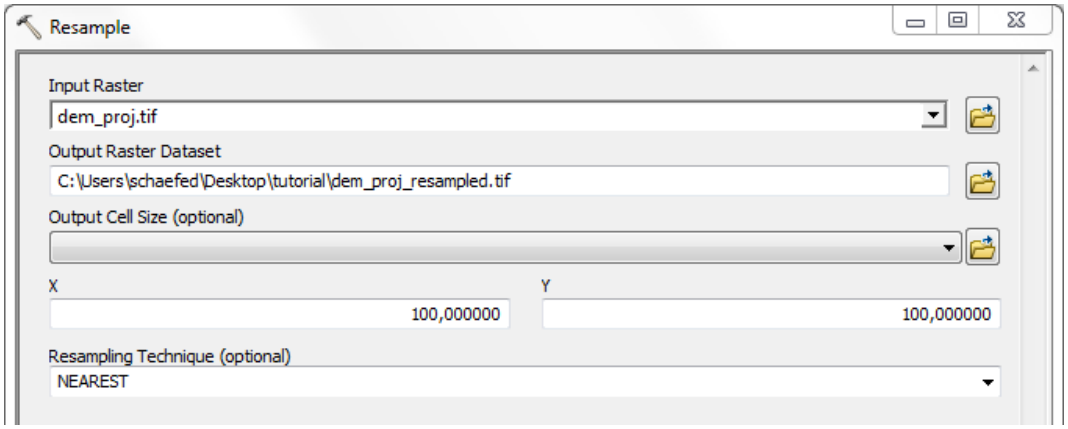


Figure 3.4: System Toolboxes -> Data Management Tools -> Raster -> Raster Processing -> Resample

- 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.

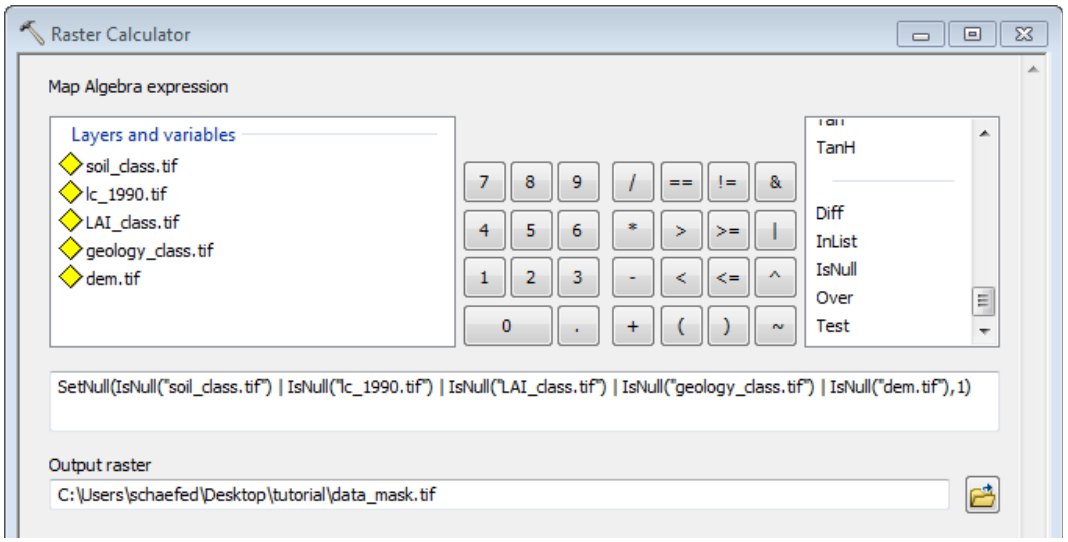


Figure 3.5: System Toolboxes -> Spatial Analyst Tools -> Map Algebra -> Raster Calculator

- 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

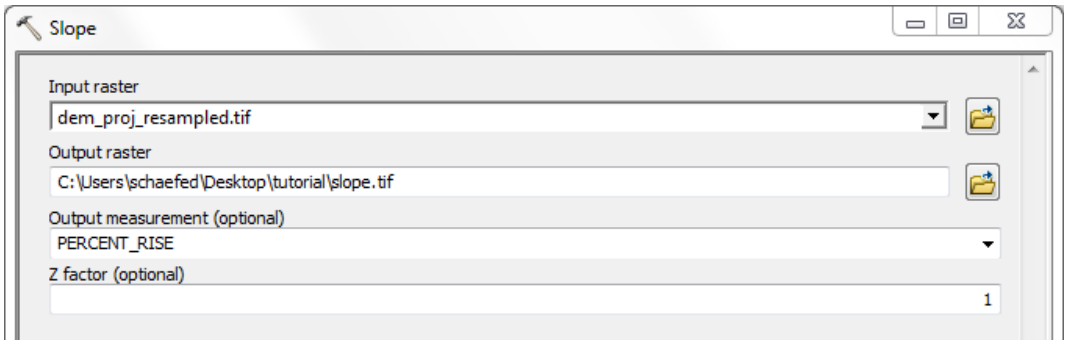


Figure 3.6: System Toolboxes -> Spatial Analyst Tools -> Surface -> Slope

3.5.3 Aspect map

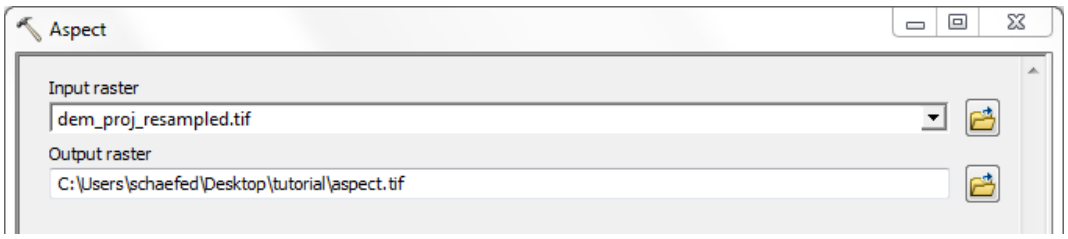


Figure 3.7: System Toolboxes -> Spatial Analyst Tools -> Surface -> Aspect

3.5.4 Fill DEM sinks

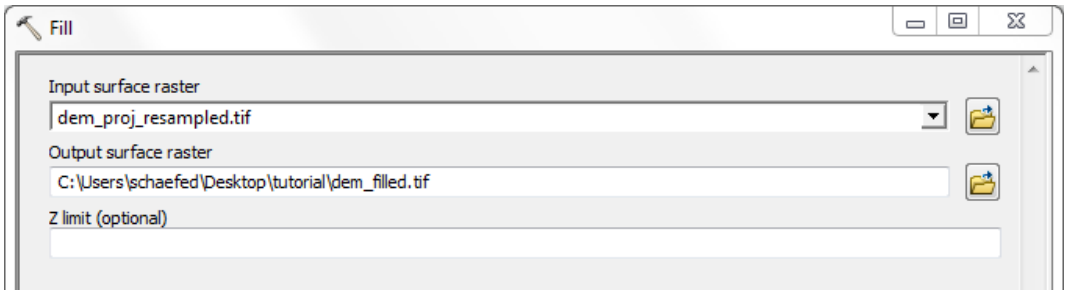


Figure 3.8: System Toolboxes -> Spatial Analyst Tools -> Hydrology -> Fill

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

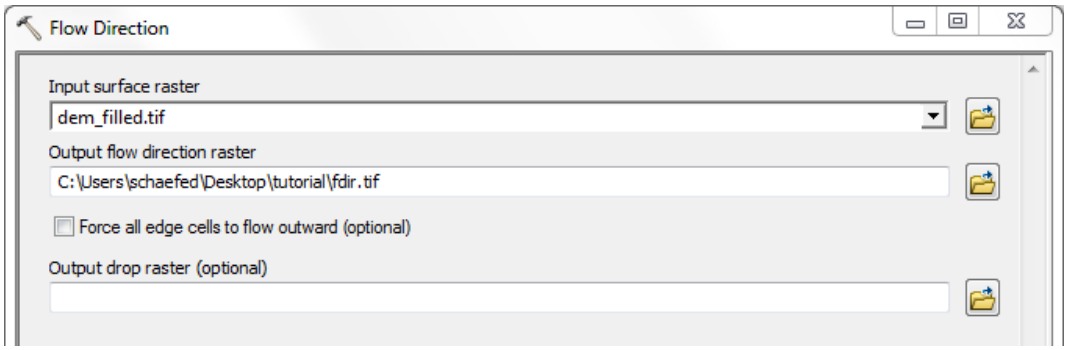


Figure 3.9: System Toolboxes -> Spatial Analyst Tools -> Hydrology -> Flow Direction

- Flow Accumulation

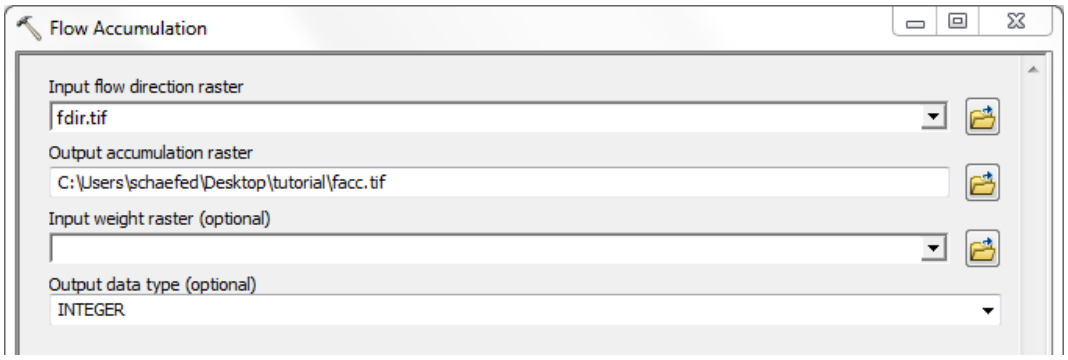


Figure 3.10: System Toolboxes -> Spatial Analyst Tools -> Hydrology -> Flow Accumulation

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.

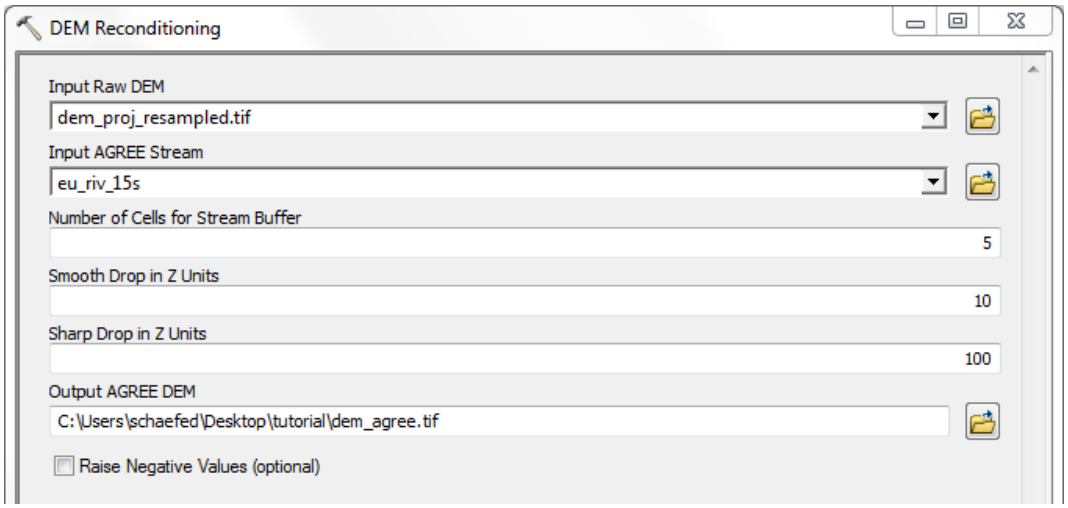


Figure 3.11: System Toolboxes -> Arc Hydro Tools -> Terrain Preprocessing -> DEM Reconditioning

- Fill the sinks in the resulting reconditioned DEM

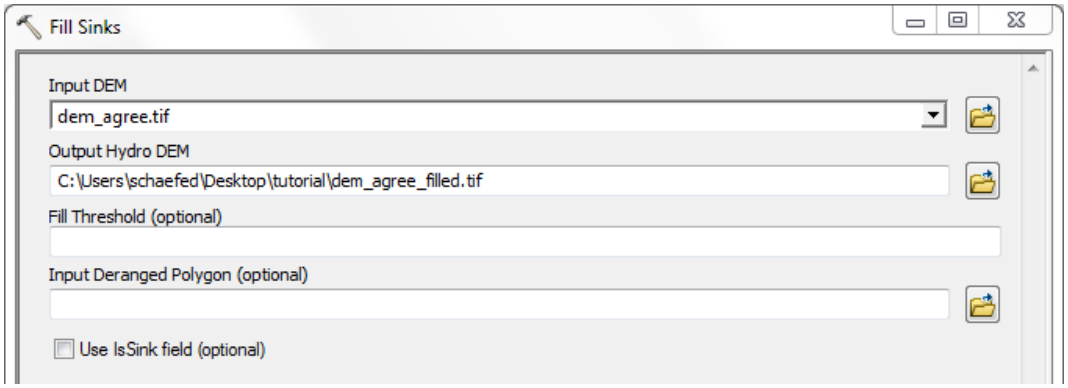


Figure 3.12: System Toolboxes -> Arc Hydro Tools -> Terrain Preprocessing -> Fill Sinks

- Calculate Flow Direction

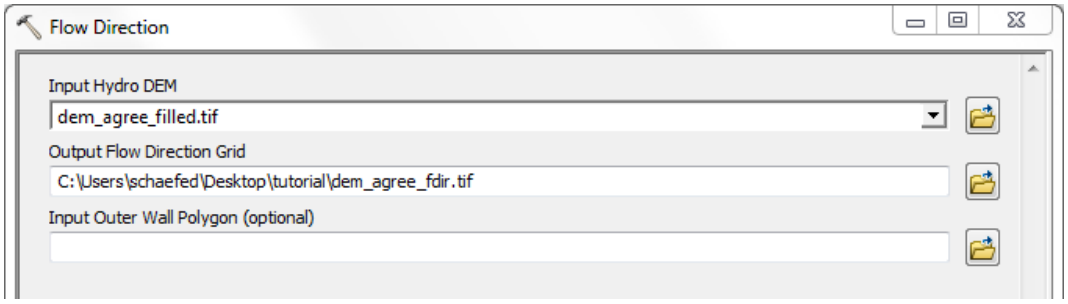


Figure 3.13: System Toolboxes -> Arc Hydro Tools -> Terrain Preprocessing -> Flow Direction

- Create a Flow Accumulation map

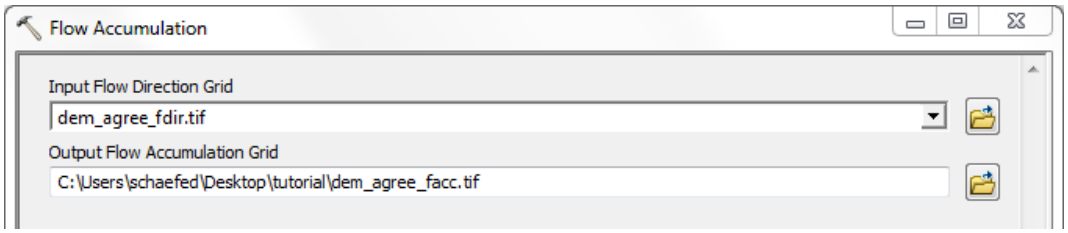


Figure 3.14: System Toolboxes -> Arc Hydro Tools -> Terrain Preprocessing -> Flow Accumulation

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.

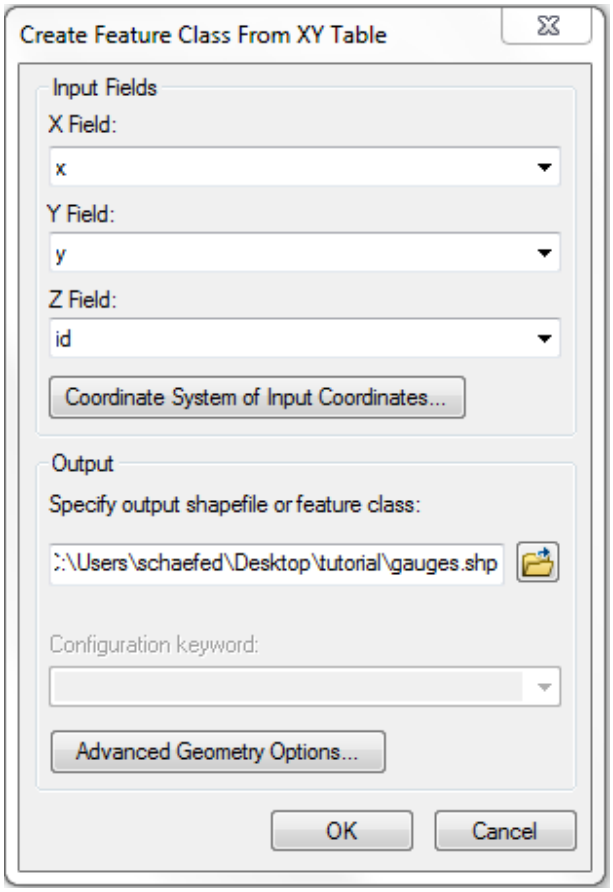


Figure 3.15: Right click on the table in Arc Catalog - > Create Feature Class -> From XY Table

- 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).

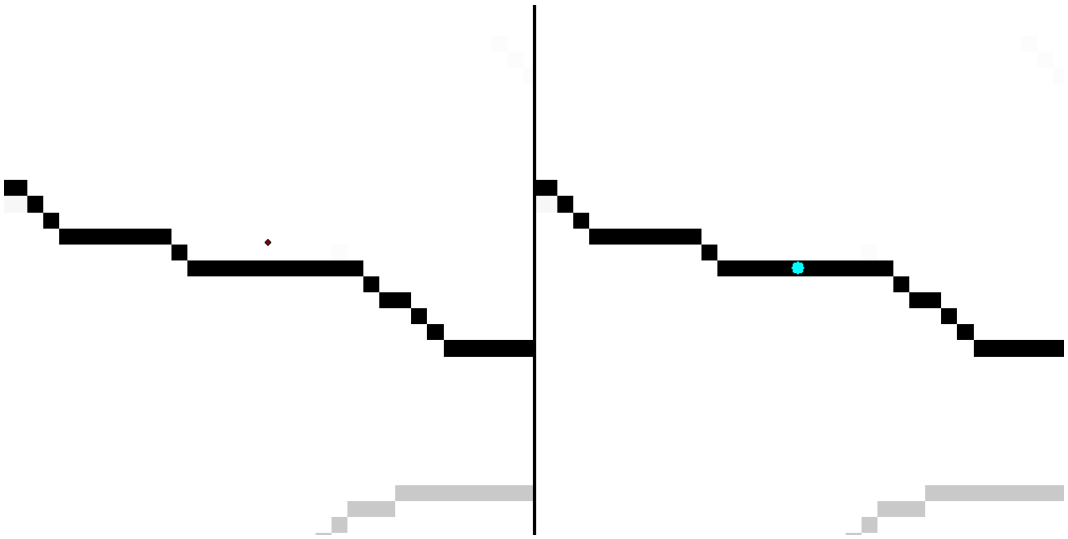


Figure 3.16: Mismatching gauges (left) should be corrected (right)

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

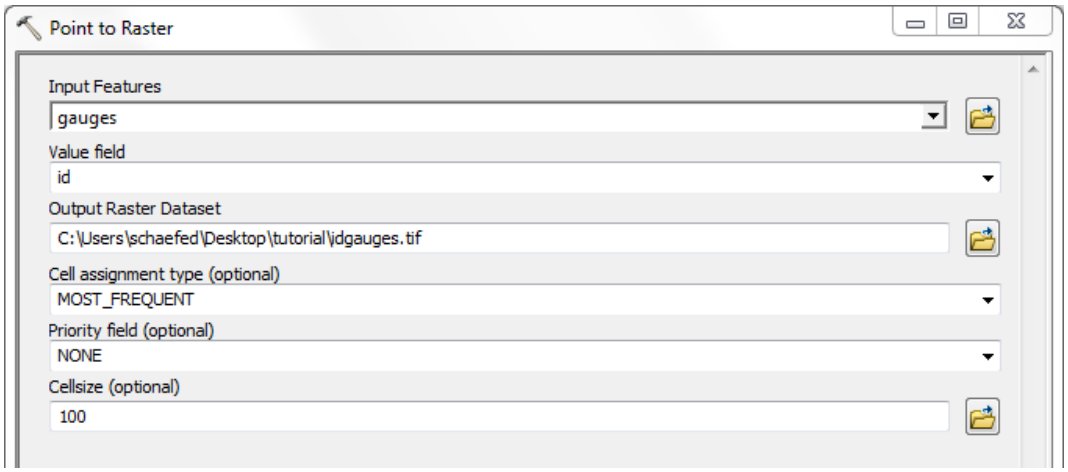


Figure 3.17: System Toolboxes -> Conversion Tools -> To Raster -> Point to Raster

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

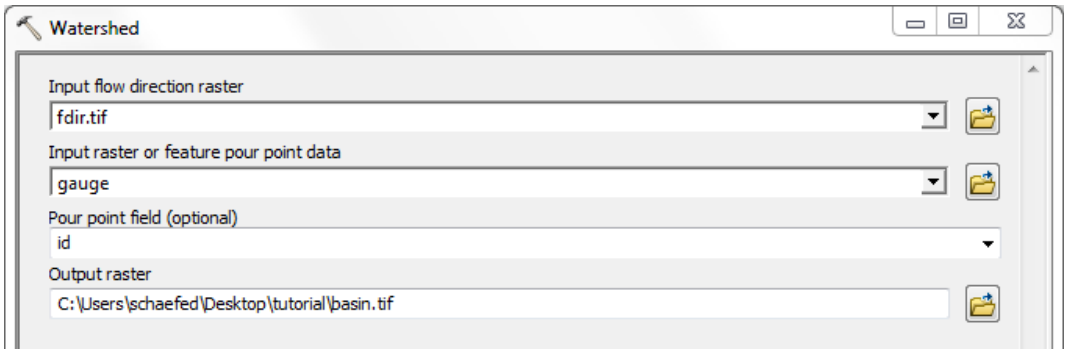


Figure 3.18: System Toolboxes -> Spatial Analyst -> Hydrology -> Watershed

3.5.8 Mask the datasets

Mask all mHM input raster files with the created watershed mask

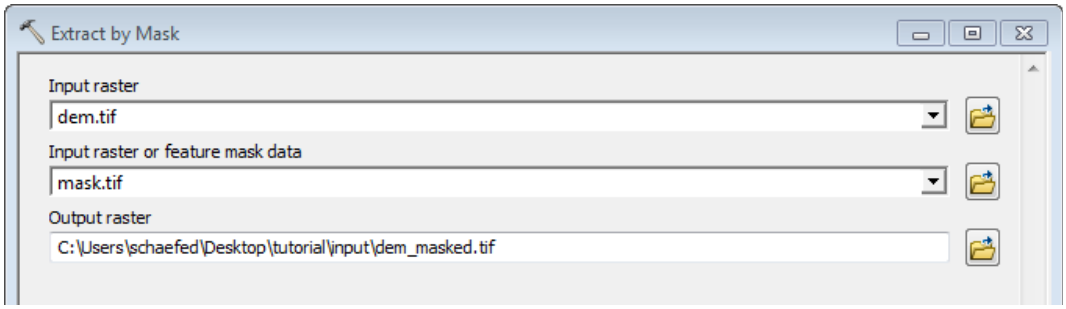


Figure 3.19: System Toolboxes -> Spatial Analysis Tools > Extraction > Extract by Mask

3.5.9 Write the ascii grids

Convert the processed and masked raster maps into ASCII files

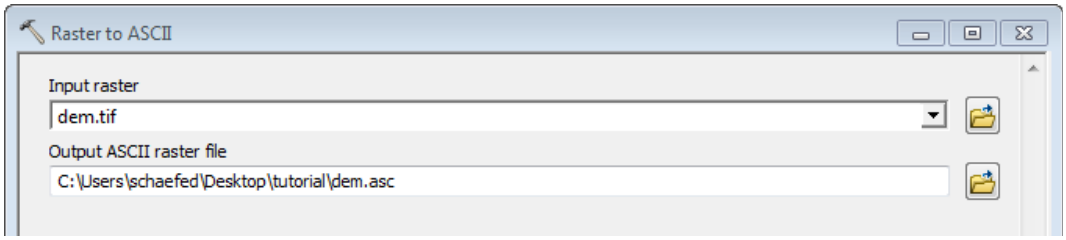


Figure 3.20: System Toolboxes -> Conversion Tools -> From Raster -> Raster to ASCII

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
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'
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
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 NCVIEW 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 NASA'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/↩ HTML/ 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)

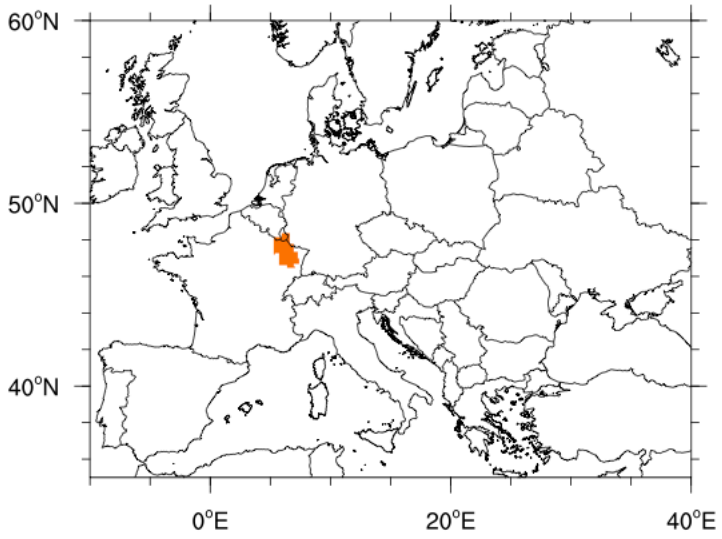


Figure 7.1: Location of the Moselle River basin upstream of Perl within the European domain

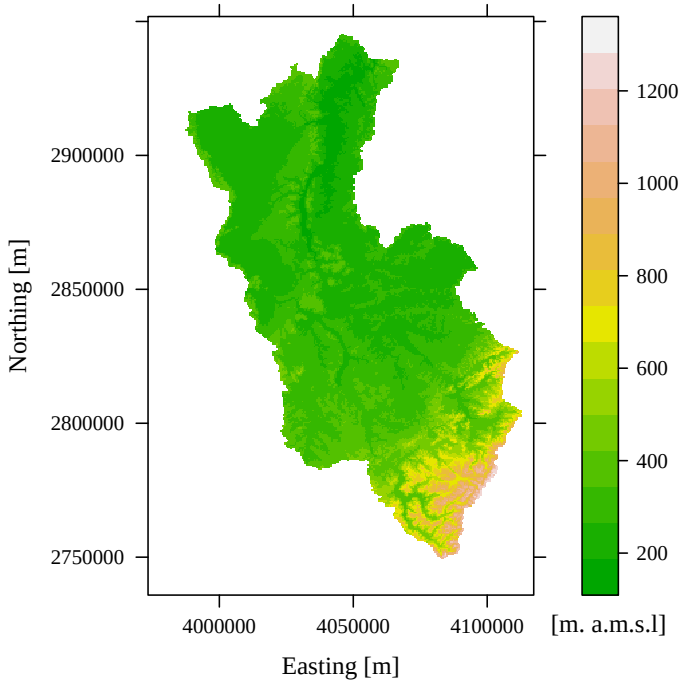


Figure 7.2: Digital elevation model for the Moselle River basin upstream of Perl

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

10.1 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

Install Instruction

The section 'Dependencies' lists the general requirements for the installation. The section 'System-dependent installation instructions' gives some instructions on how to install these dependencies on Windows, some Linux-distributions and soon MacOS.

The section 'Specific setups' can be skipped in most cases. If you are using a module system or MacOS there are some additional hints for the installation process. It might be helpful to first read the section 'Installation'. The section 'Specific setups' then helps to adjust the commands for a build on the specific system.

The section 'Installation' then is a step-by-step guide to install mHM in the command line.

11.1 Dependencies

For Windows, some Linux distributions and soon also MacOS specific installation instructions for the following list can be found below.

- a fortran compiler
- make (a tool to compile a program)
- cmake (version >= 3.5) (a tool to create a system-dependent makefile)
- fitting netcdf-fortran libraries (libraries for the use of the data format netcdf on which mhm depends)
- (optional, but makes things much easier) git

Git is a version-control system. If you want to contribute to a project, it is highly recommended to use Git. You can use Git to download (clone) the project to your local pc and have a look at the history or synchronize it without copying the whole repository again. You can also download the project folder without Git, but this would not allow you to pull updates from and push changes to our repository.

11.2 System-dependent installation instructions

Windows:

Cygwin is an environment with a terminal that allows to compile and run programs of Unix-like systems. You can find further instructions to install cygwin on the webpage, as well as instructions on how to install further dependencies after the installation.

After the installation of cygwin and its dependencies mHM will be installed using cygwin. All commands and the execution of mHM only run in that environment.

Install cygwin by executing the cygwin setup and choose the following dependencies:

- [ ] gcc-fortran (the fortran compiler)
- [ ] make
- [ ] cmake (version >= 3.12)
- [ ] libnetcdf-fortran-devel
- [ ] libhdf5-devel
- [ ] libgfortran
- [ ] gfortran

While installing cygwin you will have to choose a mirror. A mirror is a server on the internet where the files for the installation come from. Choose any server located near your city and when in doubt, choose the first one in the list. In the next step you can find all available packages provided by cygwin, set the view to "full". In the search panel you can filter the packages by the dependencies listed above (e.g. make). When you choose a version, the newest one is usually a good choice if not marked as experimental.

Note for UFZ members: Install cygwin locally (on your own computer), do not choose a location on the network for the installation.

Some cygwin versions create a new home directory for you. You may check e.g. here:

```
C:\cygwin64\home\$username
```

As from December 2019, step-by-step guidelines, how to install all cygwin libraries can be viewed in this youtube video created by Mehmet Cüneyd Demirel (Istanbul Technical University).

Step-by-step mHM compilation in CYGWIN platform

1) Change directory to mHM folder. Use single quote e.g. 'D:/mhm-v5.11.1/' if there is space in the path. cd mhm

2) Make a sub-directory inside mHM folder e.g. build mkdir build

3) Change directory to build subfolder cd build

4) Execute `cmake` with the path to the Git source directory as parameter. (If you followed the instructions above, the path is `..`) cmake ..

Other `cmake` options:

To avoid memory issues, allocate stack memory during cmake

```
cmake -DCMAKE_Fortran_FLAGS="-Wl,--stack,12485760" ..
```

If you will run mHM in parallel using OpenMP then you will need Microsoft MPI installed in your PC. Search for "Download Microsoft MPI" on internet.

Then use cmake option below. Note that memory dump is a common issue for cygwin users when compiling with OpenMP. For memory allocation please also use this line below.

```
cmake -DCMAKE_Fortran_FLAGS="${CMAKE_Fortran_FLAGS} -Wl,--stack,12485760" -DCMAKE_WITH_OpenMP=ON -DCMAKE_BUILD
```

4) Execute `make` make

5) If all went well then mhm.exe must be created inside build folder. If you are in build folder then copy mhm.exe to upper folder. cp ./mhm.exe ..

6) Change directory to upper level and then call mhm cd .. ./mhm

Ubuntu, Mint and other apt-get based systems with matching repositories

```
sudo apt-get install git # (optional)
sudo apt-get install gfortran netcdf-bin libnetcdf-dev libnetcdff-dev cmake
```

Archlinux

```
sudo pacman -S git # (optional)
sudo pacman -S gcc-libs netcdf-fortran cmake
```

Module systems

If you are on a module system, load the modules gcc or intel depending on your favorite compiler. Then, load the modules netcdf-fortran and cmake.

These modules will have system specific names, environments, etc. You may use `module spider` to find the right packages and the right dependencies, potentially use corresponding wiki pages.

On eve (the cluster at the UFZ)

A set of load-scripts is provided in `moduleLoadScripts`, to load all need modules for specifc compilers:

- GNU 7.3 compiler (`foss/2018b` Toolchain):
  `source moduleLoadScripts/eve.gcc73`

  or (MPI support)
  `source moduleLoadScripts/eve.gcc73MPI`
- GNU 8.3 compiler (`foss/2019b` Toolchain):
  `source moduleLoadScripts/eve.gcc83`

  or (MPI support)
  `source moduleLoadScripts/eve.gcc83MPI`
- Intel 18 compiler (`iomkl/2018b` Toolchain):
  `source moduleLoadScripts/eve.intel18`

  or (MPI support)
  `source moduleLoadScripts/eve.intel18MPI`
- Intel 20 compiler (`iomkl/2020a` Toolchain):
  `source moduleLoadScripts/eve.intel20`

  or (MPI support)
  `source moduleLoadScripts/eve.intel20MPI`
- NAG 6.2 compiler:
  `source moduleLoadScripts/eve.nag62`

Then you can compile mHM with cmake. We prepared a set of scripts, to automatize the build and compilation process to generate an executable in the root directory with the following naming scheme:

- Release version `mhm`:
  `source CI-scripts/compile`
- Debug version `mhm_debug`:
  `source CI-scripts/compile_debug`
- Release version with MPI support `mhm_mpi`:
  `source CI-scripts/compile_MPI`
- Debug version with MPI support `mhm_mpi_debug`:
  `source CI-scripts/compile_MPI_debug`
- Release version with OpenMP support `mhm_openmp`:
  `source CI-scripts/compile_OpenMP`
- Debug version with OpenMP support `mhm_openmp_debug`:
  `source CI-scripts/compile_OpenMP_debug`

MacOS

∗(to be added)∗

11.3 Specific setups

The following hints can replace the step `cmake ..` in the installation instruction.

You can skip this part and continue with "Installation", if you do not have a module system setup (like on clusters) or if you have not installed all packages with a package manager, such as cygwin or apt-get.

Module systems

If you are okay with loading the needed modules as described in the section 'Module systems' above before executing the program you can skip this step and continue with 'Installation'.

The executable can be built in such a way that it does not require any modules to be loaded. The module system, though, adds system paths in the backround the user should not care about too much, so the setup is a workaround. (This would be the case with any other building tool aswell.) It should be stable, anyway.

In case you want to have a module-independent build, instead of just executing `cmake ..`, either run

```
cmake -DCMAKE_BUILD_MODULE_SYSTEM_INDEPENDENT=ON ..
```

or

```
cmake -C ../CMakeCacheFiles/eve ..
```

or change the variable `CMAKE_BUILD_MODULE_SYSTEM_INDEPENDENT` with `ccmake` to `ON` after running `cmake ..`.

Non-standard locations for the netcdf-library (e.g. standard setup Macs in CHS):

Find the location of the `nf-config` file, for example by using:

```
find / -iname "*nf-config*" 2>/dev/null
```

This searches the root directory `/` for a file with a name containing the string "nf-config" (case-insensitive). It writes error messages like "permission denied" into the void.

Then, instead of running `cmake ..` if not using the standard compiler, set the fortran compiler variable to the desired compiler, e.g.

```
export FC=gfortran
```

then either run

```
cmake -DCMAKE_NETCDF_DIR=/path/to/nf-config/of/used/compiler
```

or copy the file `specificSetup` to some other place:

```
cp ../CMakeCacheFiles/specificSetup .
```

However, in case you want to keep it, you should choose a place outside the build repository. Edit the file as follows: add the path to your `nf-config` file, and after editing, run:

```
cmake -C specificSetup ..
```

or change the variable `CMAKE_NETCDF_DIR` to the path to the `nf-config` file with `ccmake` after running `cmake ..`.

11.4 Installation

1. Change to a directory where you want to store the source code.
2. Clone the corresponding mHM repository into a folder, either using Git (if installed):

   ```
   git clone --recurse-submodules https://git.ufz.de/mhm/mhm.git mhm/
   ```

   for cloning it into a folder `mhm`, or download and unpack it using the download link on https://git.ufz.de/mhm/mhm (the cloud symbol with the arrow on it).
3. Create and change to a build directory where you want to store the build, e.g. inside the Git source directory

   ```
   cd mhm
   mkdir build
   ```

   Change into the build directory:

   ```
   cd build
   ```

4. Generate a system-dependent makefile

   Execute `cmake` with the path to the Git source directory as parameter. (If you followed the instructions above, the path is `..`)

   cmake ..

   If everything worked well a Makefile was created with the corresponding paths.

   Note: have a look at "Specific setups" above in case you are using module systems, or when the netcdf libraries are not located where the package manager usually installs libraries, or when they are not saved in environment variables (i.e., classical MacOS setups at CHS).

5. Make the build:
   Execute make:

   ```
   make
   ```

   If this also worked fine, an executable was created, which has to be moved or copied to the Git source directory.
6. Execute the file:

   ```
   cd ..
   cp build/mhm .
   ```

   On Windows the executable is called `mhm.exe` instead of `mhm`. In that case instead of `cp build/mhm .` execute

   ```
   cp build/mhm.exe .
   ```

   Now you might execute mHM:

   ```
   ./mhm
   ```

   Note concerning the development of the cmake setup: one could automatically link the executable with the `cmake` code inside the Git source directory which is not done for two reasons:

- *The executable depends on your local system, so it should never be commited and pushed to other users. Nothing should be built inside the source directory automatically.*
- *The directory where mHM is executed usually is not the source directory but the directory where you want to run your tests. In case of the test setup it is the same, usually it is not.*

11.5 Building Realease or Debug versions

If you want to set up specific versions of the build, you can create different folders for that. In case a release and a debug version need to be set up, it makes sense to create two folders named `debug` and `release`.

```
mkdir release

mkdir debug
```

inside the `release` folder one would execute

```
cmake -DCMAKE_BUILD_TYPE=Release ..
```

and inside the `debug` folder

```
cmake -DCMAKE_BUILD_TYPE=Debug ..
```

Executing

```
make
```

in the corresponding folder would then always result in a release build or respectively in a debug build.

11.6 Troubleshooting

cmake is far from being my main task, so it will probably take a while until I can track a problem. Nonetheless, I would be happy having bug reports. Feel free to write an email to maren.kaluza@ufz.de if there are any bugs, remarks or questions.

MacOS, homebrew

On brew/homebrew setup MacOS systems a working nf-config is not yet available at this time. Execute:

```
nf-config --all
```

and if it says something like "is not implemented yet" the issue is not solved yet. But it is on my tracklist.

cygwin

If libraries are not found, the problem can be:

- you accidentally tried to use commands within the cmd shell and not within the cygwin shell
- your cygwin setup might be broken. Try deinstalling, following the instructions https://cygwin.com/faq/faq.html#faq.setup.uninstall-service, and reinstalling again. During reinstallation only install the required dependencies.

Chapter 12

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., Acuña, 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 13

mHM Release Notes

13.1 mHM v5.11.2 (Jul 2021)

Enhancements

- documentation modernized with `doxygen-awesome-css` (!86)
- cmake update to be able to install mHM (`cmake --install`) (!85)
- added pFUnit tests thanks to Nicola Döring (!76)
- link to a new `YouTube tutorial` for compiling mHM with cygwin by Mehmet Cüneyd Demirel added to the documentation (!74)
- NetCDF output: add deflate and precision option to namelists (!73)
- refactor cmake workflow (!72)

Bugfixes

- fixed: `mrm` tried to write output even if routing was switched off (!82)
- unreachable `else` branch in `feddes_et_reduction` removed (!77)
- unnecessary `inout` variable intent in `soil_moisture` removed (!77)

13.2 mHM v5.11.1 (Mar 2021)

Enhancements

- added compile information for cygwin (!68)

Bugfixes

- removed note about mHM 5.10 from the README
- smhorizon: tmp_rootfraccoef was corrected directly if it is not between 0 and 1, but actually FCnorm should always be between 0 and 1 (!67)

13.3 mHM v5.11 (Feb 2021)

Experimental Features

- river temperature routing was implemented in an alpha version 0.1 (!37)
  - this feature is in an experimental stage and should not be considered stable!

Enhancements

- introduced central version files `version.txt` and `version_data.txt` (!51)
- added Feddes and global FC dependency on root fraction coef. at SM process(3)=4 (!43)
- Online documentation generated with doxygen: https://mhm.pages.ufz.de/mhm/develop/ (!44)
- CI/CD with GitLab Runner (!11, !13, !14, !28, !32, !48, !50)
  - building on EVE for multiple compiler (GNU 7.3/8.3, Intel 18/19, NAG 6.2)
  - building debug/release serial/parallel
  - memcheck with valgrind
  - running all check-cases with all compiled versions
  - calculation of coverage
  - new checking script `run_mhm_checks.py`
- the domain loop is now parallelized with MPI
- objective function for boxcox-transformed streamflow
- post processing script for probabilistic forecasts
- different module load scripts for EVE
- Objective function from separate mhm calls (!7)
- new data type for simulated gridded optidata (!10)
- new datatype datetimeinfo (!16)
- added module `mo_os` to check files and directories (!41, !57)

Changes

- internal: "basin" renamed to "domain"
- TWS input file changed from ascii to netCDF (!9)
- Switched to cell wise kge of et and tws in opti_function 33 (!12)
- restart files are now given by name (!34)
- removed mRM standalone and statically integrate mRM into mHM (!53)
- removed the old makefile and legacy checking scripts (!55)
- minimal Cmake version is now `3.12` (!58)

Bugfixes

- Finalparam.nml is now written with specific format (Intel/GNU compatibility) (#40)
- FinalParam.nml routing section bug fixed (#49, !25)
- dirEvapotranspiration is now allocated before writing
- cmake: netcdf link flags where separated by ";"
- sharing of L0 domain now working
- added L1_jarvis_thresh_c1 to restart file for process id 2 AND 3 (#29, !15)
- allowing higher routing resolution than hydrology (!21)
- domainID not set correctly for mRM if restart is activated (!30)
- mHM states_fluxes netCDF output was curvilinear even if coordinate system is set to regular latlon (#98, !31)
- missmatch in messages about written mhm fluxes (!42)
- Fixing wrongly matched IDs from L1 to L11 when routing resolution (L11) is finer than L1 resolution (!45)
- The length in net_startup was only cut in case there are less then 2 lengths (!46)
- corrected unit attributes for lat lon variables (!47)
- Allow run mHM and mRM without any observed gauge for processCase(8) = 2 / 3 (#27, !52)

13.4 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:

1. Adaptive routing does not allow to run without at least 1 gauge specification
2. Incompatibility of Finalparam.nml format between Intel and GNU
3. If ProcessOption(3) is set to 3 and optimization is activated, the created FinalParam.nml misses the header for the namelist of the soil moisture parameters.
4. Land cover scenes cannot be changed between the run generating the restart file and the run using the restart file. This causes unpredictable behaviour by the model.
5. Simulation period must span overall land cover scenes specified in the namelist.
6. Cut-off for link length is calculated with missing values, but those should be neglected.
7. Using a higher routing resolution than hydrology resolution may cause segmentation faults because mapping from L1id on L11 is not working correctly
8. If ProcessOption(5) is set to -1 and optimization is activated, the created FinalParam.nml misses the header for the namelist of the PET process.

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.

13.5 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.

13.6 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.

13.7 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 *GEUS*.

  - Jarvis (1989, J. Hydrol.) evapotranspiration reduction (`processCase(3)=3`) – implemented by *GEUS*.

- Use local, monthly LAI climatology instead of look-up table, i.e., LAI_classdefinition.txt (`timeStep_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.

13.8 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.

13.9 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.

13.10 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`.

13.11 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)`).

13.12 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.**

13.13 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.

13.14 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 14

Modules Index

14.1 Modules List

Here is a list of all modules with brief descriptions:

dummy_mpr . . . 83
mo_canopy_interc
    Canopy interception . . . 83
mo_check . . . 84
mo_common_constants
    Provides constants commonly used by mHM, mRM and MPR . . . 85
mo_common_datetime_type . . . 89
mo_common_file
    Provides file names and units for mRM . . . 90
mo_common_functions
    Provides small utility functions used by multiple parts of the code (mHM, mRM, MPR) . . . 91
mo_common_mhm_mrm_file
    Provides file names and units for mHM . . . 92
mo_common_mhm_mrm_mpi_tools
    Tools for MPI communication that are mHM or mRM specific . . . 94
mo_common_mhm_mrm_read_config
    Reading of main model configurations . . . 95
mo_common_mhm_mrm_restart
    TODO: add description . . . 99
mo_common_mhm_mrm_variables
    Provides structures needed by mHM, mRM and/or mpr . . . 100
mo_common_read_config
    Reading of main model configurations . . . 107
mo_common_read_data
    TODO: add description . . . 111
mo_common_restart
    TODO: add description . . . 113
mo_common_variables
    Provides structures needed by mHM, mRM and/or mpr . . . 116
mo_file
    Provides file names and units for mHM . . . 124
mo_global_variables
    Global variables ONLY used in reading, writing and startup . . . 127
mo_grid
    TODO: add description . . . 141
mo_init_states
    Initialization of all state variables of mHM . . . 148

mo_meteo_forcings
Prepare meteorological forcings data for mHM . . . 150
mo_mhm
Call all main processes of mHM . . . 157
mo_mhm_constants
Provides mHM specific constants . . . 163
mo_mhm_eval
Runs mhm with a specific parameter set and returns required variables, e.g. runoff . . . 168
mo_mhm_read_config
Reading of main model configurations . . . 171
mo_mpr_constants
Provides MPR specific constants . . . 173
mo_mpr_eval
Runs MPR and writes to global effective parameters . . . 181
mo_mpr_file
Provides file names and units for mRM . . . 183
mo_mpr_global_variables
Global variables for mpr only . . . 189
mo_mpr_pet
TODO: add description . . . 200
mo_mpr_read_config
Read mpr config . . . 206
mo_mpr_restart
Reading and writing states, fluxes and configuration for restart of mHM . . . 207
mo_mpr_runoff
Multiscale parameter regionalization for runoff generation . . . 211
mo_mpr_smhorizons
Setting up the soil moisture horizons . . . 214
mo_mpr_soilmoist
Multiscale parameter regionalization (MPR) for soil moisture . . . 216
mo_mpr_startup
Startup procedures for mHM . . . 223
mo_mrm_constants
Provides mRM specific constants . . . 228
mo_mrm_file
Provides file names and units for mRM . . . 229
mo_mrm_global_variables
Global variables for mRM only . . . 235
mo_mrm_init
Wrapper for initializing Routing . . . 250
mo_mrm_mpr
Perform Multiscale Parameter Regionalization on Routing Parameters . . . 258
mo_mrm_net_startup
Startup drainage network for mHM . . . 261
mo_mrm_objective_function_runoff
Objective Functions for Optimization of mHM/mRM against runoff . . . 276
mo_mrm_pre_routing
Performs pre-processing for routing for mHM at level L11 . . . 306
mo_mrm_read_config
Read mRM config . . . 311
mo_mrm_read_data
This module contains all routines to read mRM data from file . . . 314
mo_mrm_restart
Restart routines . . . 319
mo_mrm_riv_temp_class
Class for the river temperature calculations . . . 323
mo_mrm_river_head . . . 333

mo_mrm_routing
Performs runoff routing for mHM at level L11 . . . 338
mo_mrm_signatures
Module with calculations for several hydrological signatures . . . 343
mo_mrm_write
Write of discharge and restart files . . . 352
mo_mrm_write_fluxes_states
Creates NetCDF output for different fluxes and state variables of mHM . . . 358
mo_multi_param_reg
Multiscale parameter regionalization (MPR) . . . 366
mo_neutrons
Models to predict neutron intensities above soils . . . 378
mo_objective_function
Objective Functions for Optimization of mHM . . . 386
mo_optimization
Wrapper subroutine for optimization against runoff and sm . . . 409
mo_pet
Module for calculating reference/potential evapotranspiration [mm d-1] . . . 410
mo_prepare_gridded_lai
Prepare daily LAI fields (e.g., MODIS data) for mHM . . . 417
mo_read_latlon
Reading latitude and longitude coordinates for each domain . . . 420
mo_read_lut
Routines reading lookup tables (lut) . . . 421
mo_read_nc
Reads forcing input data . . . 424
mo_read_optional_data
Read optional data for mHM calibration . . . 429
mo_read_spatial_data
Reads spatial input data . . . 431
mo_read_timeseries
Routines to read files containing timeseries data . . . 434
mo_read_wrapper
Wrapper for all reading routines . . . 435
mo_restart
Reading and writing states, fluxes and configuration for restart of mHM . . . 438
mo_runoff
Runoff generation for the unsaturated zone, saturated zone (or groundwater zone), and runoff accumulation . . . 443
mo_snow_accum_melt
Snow melting and accumulation . . . 446
mo_soil_database
Generating soil database from input file . . . 448
mo_soil_moisture
Soil moisture of the different layers . . . 450
mo_spatial_agg_disagg_forcing
Spatial aggegation or disaggregation of meteorological input data . . . 454
mo_startup
Startup procedures for mHM . . . 456
mo_template
Template for future module developments . . . 460
mo_temporal_disagg_forcing
Temporal disaggregation of daily input values . . . 462
mo_upscaling_operators
Module containing upscaling operators . . . 467
mo_write_ascii
Module to write ascii file output . . . 474

mo_write_fluxes_states
Creates NetCDF output for different fluxes and state variables of mHM 477

Chapter 15

Data Type Index

15.1 Data Types List

Here are the data types with brief descriptions:

mo_common_mhm_mrm_restart::check_consistency_element . . . 487
mo_common_datetime_type::datetimeinfo . . . 488
mo_common_variables::domain_meta . . . 491
mo_mrm_global_variables::domaininfo_mrm . . . 493
mo_mrm_global_variables::gaugingstation . . . 495
mo_common_variables::grid . . . 496
mo_common_variables::gridremapper . . . 499
mo_template::mean
The average . . . 500
mo_mrm_write_fluxes_states::outputdataset . . . 502
mo_write_fluxes_states::outputdataset . . . 505
mo_mrm_write_fluxes_states::outputvariable . . . 509
mo_write_fluxes_states::outputvariable . . . 514
mo_common_variables::period . . . 518
mo_read_spatial_data::read_spatial_data_ascii
Reads spatial data files of ASCII format . . . 520
mo_mrm_riv_temp_class::riv_temp_type
This is a container to define the river temperature routing in the current time step . . . 523
mo_mpr_global_variables::soiltype . . . 538
mo_spatial_agg_disagg_forcing::spatial_aggregation
Spatial aggregation of meterological variables . . . 541
mo_spatial_agg_disagg_forcing::spatial_disaggregation
Spatial disaggregation of meterological variables . . . 542
mo_mpr_restart::unpack_field_and_write
TODO: add description . . . 543
mo_restart::unpack_field_and_write
TODO: add description . . . 545

Chapter 16

File Index

16.1 File List

Here is a list of all files with brief descriptions:

src/common/mo_check.f90
Input checking routines . . . 549
src/common/mo_common_constants.f90 . . . 550
src/common/mo_common_datetime_type.f90 . . . 550
src/common/mo_common_file.f90 . . . 551
src/common/mo_common_functions.f90 . . . 551
src/common/mo_common_read_config.f90 . . . 551
src/common/mo_common_read_data.f90 . . . 552
src/common/mo_common_restart.f90 . . . 552
src/common/mo_common_variables.f90 . . . 552
src/common/mo_grid.f90 . . . 553
src/common/mo_read_latlon.f90 . . . 554
src/common/mo_read_nc.f90 . . . 554
src/common/mo_read_spatial_data.f90 . . . 555
src/common/mo_read_timeseries.f90 . . . 555
src/common/mo_template.f90 . . . 555
src/common_mHM_mRM/mo_common_mHM_mRM_file.f90 . . . 556
src/common_mHM_mRM/mo_common_mHM_mRM_MPI_tools.f90 . . . 556
src/common_mHM_mRM/mo_common_mHM_mRM_read_config.f90 . . . 557
src/common_mHM_mRM/mo_common_mHM_mRM_restart.f90 . . . 557
src/common_mHM_mRM/mo_common_mHM_mRM_variables.f90 . . . 557
src/common_mHM_mRM/mo_optimization.f90 . . . 558
src/mHM/mhm_driver.f90 . . . 558
src/mHM/mo_canopy_interc.f90 . . . 561
src/mHM/mo_file.f90 . . . 562
src/mHM/mo_global_variables.f90 . . . 562
src/mHM/mo_init_states.f90 . . . 564
src/mHM/mo_meteo_forcings.f90 . . . 564
src/mHM/mo_mhm.f90 . . . 564
src/mHM/mo_mhm_constants.f90 . . . 565
src/mHM/mo_mhm_eval.f90 . . . 566
src/mHM/mo_mhm_read_config.f90 . . . 566
src/mHM/mo_neutrons.f90 . . . 566
src/mHM/mo_objective_function.f90 . . . 567
src/mHM/mo_pet.f90 . . . 567
src/mHM/mo_read_optional_data.f90 . . . 568
src/mHM/mo_restart.f90 . . . 568
src/mHM/mo_runoff.f90 . . . 569

src/mHM/mo_snow_accum_melt.f90 . . . 569
src/mHM/mo_soil_moisture.f90 . . . 569
src/mHM/mo_spatial_agg_disagg_forcing.f90 . . . 570
src/mHM/mo_startup.f90 . . . 570
src/mHM/mo_temporal_disagg_forcing.f90 . . . 570
src/mHM/mo_write_ascii.f90 . . . 571
src/mHM/mo_write_fluxes_states.f90 . . . 571
src/MPR/mo_mpr_constants.f90 . . . 572
src/MPR/mo_mpr_eval.f90 . . . 573
src/MPR/mo_mpr_file.f90 . . . 573
src/MPR/mo_mpr_global_variables.f90 . . . 574
src/MPR/mo_mpr_pet.f90 . . . 576
src/MPR/mo_mpr_read_config.f90 . . . 576
src/MPR/mo_mpr_restart.f90 . . . 576
src/MPR/mo_mpr_runoff.f90 . . . 577
src/MPR/mo_mpr_smhorizons.f90 . . . 577
src/MPR/mo_mpr_soilmoist.f90 . . . 577
src/MPR/mo_mpr_startup.f90 . . . 578
src/MPR/mo_multi_param_reg.f90 . . . 578
src/MPR/mo_prepare_gridded_lai.f90 . . . 579
src/MPR/mo_read_lut.f90 . . . 579
src/MPR/mo_read_wrapper.f90 . . . 579
src/MPR/mo_soil_database.f90 . . . 579
src/MPR/mo_upscaling_operators.f90 . . . 580
src/MPR/mpr_driver.f90 . . . 580
src/mRM/mo_mrm_constants.f90 . . . 583
src/mRM/mo_mrm_file.f90 . . . 583
src/mRM/mo_mrm_global_variables.f90 . . . 584
src/mRM/mo_mrm_init.f90 . . . 586
src/mRM/mo_mrm_mpr.f90 . . . 586
src/mRM/mo_mrm_net_startup.f90 . . . 586
src/mRM/mo_mrm_objective_function_runoff.f90 . . . 588
src/mRM/mo_mrm_pre_routing.f90 . . . 589
src/mRM/mo_mrm_read_config.f90 . . . 589
src/mRM/mo_mrm_read_data.f90 . . . 590
src/mRM/mo_mrm_restart.f90 . . . 590
src/mRM/mo_mrm_riv_temp_class.f90 . . . 590
src/mRM/mo_mrm_river_head.f90 . . . 591
src/mRM/mo_mrm_routing.f90 . . . 592
src/mRM/mo_mrm_signatures.f90 . . . 592
src/mRM/mo_mrm_write.f90 . . . 593
src/mRM/mo_mrm_write_fluxes_states.f90 . . . 593

Chapter 17

Module Documentation

17.1 dummy_mpr Module Reference

17.2 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.

17.2.1 Detailed Description

Canopy interception.

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

Authors

Vladyslav Prykhodko

Date

Dec 2012

17.2.2 Function/Subroutine Documentation

17.2.2.1 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 TS-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 TS-1]
out	*REAL(dp) :: evap_canopy*	Real evaporation intensity from canopy[mm TS-1]

Authors

Vladyslav Prykhodko

Date

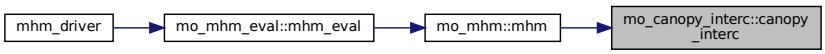
Dec 2012

Definition at line 74 of file mo_canopy_interc.f90.

References mo_common_constants::eps_dp.

Referenced by mo_mhm::mhm().

Here is the caller graph for this function:



17.3 mo_check Module Reference

Functions/Subroutines

- subroutine, public check_dir (path, text_, throwError_, tab_, text_length_)

  Check if a given directory exists.

17.3.1 Function/Subroutine Documentation

17.3.1.1 check_dir()

```
subroutine, public mo_check::check_dir (
             character(len=*), intent(in) path,
             character(len=*), intent(in), optional text_,
             logical, intent(in), optional throwError_,
             integer(i4), intent(in), optional tab_,
             integer(i4), intent(in), optional text_length_ )
```

Check if a given directory exists.

Check if a given directory exists and write out a message about it. Will also give potential information about prefixes given with the path

Authors

Sebastian Mueller

Date

Nov 2020

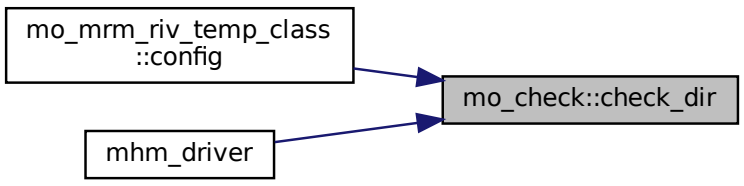
Parameters

in	*path*	input path to check
in	*text_*	text to write out
in	*throwerror_*	wheather to throw an error if folder not existing
in	*tab_*	tab-depth
in	*text_length_*	maximal text length (for aligning)

Definition at line 25 of file mo_check.f90.

Referenced by mo_mrm_riv_temp_class::config(), and mhm_driver().

Here is the caller graph for this function:



17.4 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 maxnodomains = 50_i4
- integer(i4), parameter, public maxnlcovers = 50_i4
- character(64), parameter, public soilhorizonsvarname = "L1_SoilHorizons"
- character(64), parameter, public landcoverperiodsvarname = "L1_LandCoverPeriods"
- character(64), parameter, public laivarname = "L1_LAITimesteps"

17.4.1 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

17.4.2 Variable Documentation

17.4.2.1 eps_dp

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

epsilon(1.0) in double precision

Definition at line 24 of file mo_common_constants.f90.

Referenced by mo_multi_param_reg::aerodynamical_resistance(), mo_canopy_interc::canopy_interc(), 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().

17.4.2.2 eps_sp

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

epsilon(1.0) in single precision

Definition at line 26 of file mo_common_constants.f90.

17.4.2.3 laivarname

```
character(64), parameter, public mo_common_constants::laivarname = "L1_LAITimesteps"
```

Definition at line 42 of file mo_common_constants.f90.

Referenced by mo_restart::read_restart_states(), and mo_mpr_restart::write_mpr_restart_files().

17.4.2.4 landcoverperiodsvarname

```
character(64), parameter, public mo_common_constants::landcoverperiodsvarname = "L1_LandCover↩
Periods"
```

Definition at line 41 of file mo_common_constants.f90.

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

17.4.2.5 maxnlcovers

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

Definition at line 38 of file mo_common_constants.f90.

Referenced by mo_common_read_config::common_read_config().

17.4.2.6 maxnodomains

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

Definition at line 37 of file mo_common_constants.f90.

Referenced by mo_common_mhm_mrm_read_config::common_mhm_mrm_read_config(), mo_common_read↩
_config::common_read_config(), mo_mrm_riv_temp_class::config(), mo_mhm_read_config::mhm_read_config(), mo_mpr_read_config::mpr_read_config(), and mo_mrm_read_config::mrm_read_config().

17.4.2.7 ncolpars

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

Definition at line 36 of file mo_common_constants.f90.

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

17.4.2.8 nodata_dp

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

Definition at line 30 of file mo_common_constants.f90.

Referenced by mo_mrm_river_head::avg_and_write_timestep(), mo_multi_param_reg::baseflow_param(), mo_↩
meteo_forcings::chunk_config(), mo_objective_function::create_domain_avg_et(), mo_objective_function::create↩
_domain_avg_tws(), mo_mrm_river_head::create_output(), mo_write_fluxes_states::createoutputfile(), mo_mrm↩

_write_fluxes_states::createoutputfile(), 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_pre_routing::l11_e_acc(), mo_mrm_net_startup::l11_flow_accumulation(), mo_mrm_net_startup::l11_fraction_sealed_floodplain(), mo_mrm_pre_routing::l11_meteo_acc(), mo_mrm_pre_routing::l11_runoff_acc(), mo_mrm_net_startup::l11_stream_features(), mo_meteo_forcings::meteo_forcings_wrapper(), mo_meteo_forcings::meteo_weights_wrapper(), 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_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_master(), mo_objective_function::objective_neutrons_kge_catchment_avg(), mo_objective_function::objective_q_et_tws_kge_catchment_avg(), mo_objective_function::objective_sm_kge_catchment_avg(), mo_objective_function::objective_sm_pd(), mo_objective_function::objective_subprocess(), mo_read_wrapper::read_data(), mo_common_read_data::read_dem(), mo_soil_database::read_soil_lut(), mo_read_optional_data::readoptidataobs(), mo_spatial_agg_disagg_forcing::spatial_aggregation::spatial_aggregation_3d(), mo_spatial_agg_disagg_forcing::spatial_aggregation::spatial_aggregation_4d(), mo_spatial_agg_disagg_forcing::spatial_disaggregation::spatial_disaggregation_3d(), mo_spatial_agg_disagg_forcing::spatial_disaggregation::spatial_disaggregation_4d(), mo_write_ascii::write_configfile(), mo_mrm_write::write_configfile(), mo_mrm_write::write_daily_obs_sim_discharge(), mo_mpr_restart::write_eff_params(), mo_common_restart::write_grid_info(), mo_restart::write_restart_files(), mo_write_fluxes_states::writevariableattributes(), mo_mrm_write_fluxes_states::writevariableattributes(), mo_write_fluxes_states::writevariabletimestep(), and mo_mrm_write_fluxes_states::writevariabletimestep().

17.4.2.9 nodata_i4

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

Definition at line 29 of file mo_common_constants.f90.

Referenced by mo_mrm_river_head::calc_channel_elevation(), mo_mrm_riv_temp_class::config(), 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().

17.4.2.10 p1_initstatefluxes

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

Definition at line 33 of file mo_common_constants.f90.

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().

17.4.2.11 soilhorizonsvarname

```
character(64), parameter, public mo_common_constants::soilhorizonsvarname = "L1_SoilHorizons"
```

Definition at line 40 of file mo_common_constants.f90.

Referenced by mo_restart::read_restart_states(), and mo_mpr_restart::write_mpr_restart_files().

17.5 mo_common_datetime_type Module Reference

Data Types

- type datetimeinfo

Functions/Subroutines

- subroutine datetimeinfo_init (this, iDomain)
- subroutine datetimeinfo_increment (this)
- subroutine datetimeinfo_update_lai_timestep (this)
- logical function datetimeinfo_writeout (this, timeStep_model_outputs, tt)

17.5.1 Function/Subroutine Documentation

17.5.1.1 datetimeinfo_increment()

```
subroutine mo_common_datetime_type::datetimeinfo_increment (
            class(datetimeinfo), intent(inout) this )
```

Definition at line 105 of file mo_common_datetime_type.f90.

References mo_common_mhm_mrm_variables::timestep.

17.5.1.2 datetimeinfo_init()

```
subroutine mo_common_datetime_type::datetimeinfo_init (
            class(datetimeinfo), intent(inout) this,
            integer(i4), intent(in) iDomain )  [private]
```

Definition at line 70 of file mo_common_datetime_type.f90.

References mo_common_mhm_mrm_variables::lcyearid, mo_common_mhm_mrm_variables::ntstepday, mo_common_mhm_mrm_variables::simper, and mo_common_mhm_mrm_variables::timestep.

17.5.1.3 datetimeinfo_update_lai_timestep()

```
subroutine mo_common_datetime_type::datetimeinfo_update_lai_timestep (
            class(datetimeinfo), intent(inout) this )
```

Definition at line 132 of file mo_common_datetime_type.f90.

References mo_mpr_global_variables::timestep_lai_input.

17.5.1.4 datetimeinfo_writeout()

```
logical function mo_common_datetime_type::datetimeinfo_writeout (
            class(datetimeinfo), intent(in) this,
            integer(i4), intent(in) timeStep_model_outputs,
            integer(i4), intent(in) tt )
```

Definition at line 154 of file mo_common_datetime_type.f90.

17.6 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.

17.6.1 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

17.6.2 Variable Documentation

17.6.2.1 file_config

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

file defining mHM's outputs

Definition at line 26 of file mo_common_file.f90.

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

17.6.2.2 file_dem

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

DEM input data file.

Definition at line 19 of file mo_common_file.f90.

Referenced by mo_common_read_data::read_dem().

17.6.2.3 uconfig

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

Unit for file defining mHM's outputs.

Definition at line 28 of file mo_common_file.f90.

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

17.6.2.4 udem

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

Unit for DEM input data file.

Definition at line 21 of file mo_common_file.f90.

Referenced by mo_common_read_data::read_dem().

17.6.2.5 ulcoverclass

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

Unit for LCover input data file.

Definition at line 23 of file mo_common_file.f90.

Referenced by mo_common_read_data::read_lcover().

17.7 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.

17.7.1 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

17.7.2 Function/Subroutine Documentation

17.7.2.1 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

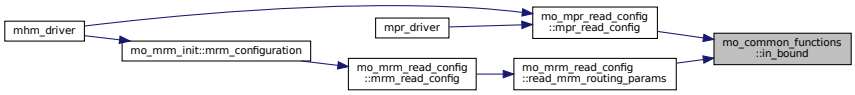
Date

Jun 2018

Definition at line 46 of file mo_common_functions.f90.

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:



17.8 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)

17.8.1 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

17.8.2 Variable Documentation

17.8.2.1 file_opti

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

file defining optimization outputs (objective and parameter set)

Definition at line 18 of file mo_common_mHM_mRM_file.f90.

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

17.8.2.2 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)

Definition at line 22 of file mo_common_mHM_mRM_file.f90.

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

17.8.2.3 uopti

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

Unit for file optimization outputs (objective and parameter set)

Definition at line 20 of file mo_common_mHM_mRM_file.f90.

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

17.8.2.4 uopti_nml

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

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

Definition at line 24 of file mo_common_mHM_mRM_file.f90.

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

17.9 mo_common_mhm_mrm_mpi_tools Module Reference

tools for MPI communication that are mHM or mRM specific

Functions/Subroutines

- subroutine, public distribute_parameterset (parameterset)
- subroutine, public get_parameterset (parameterset)

17.9.1 Detailed Description

tools for MPI communication that are mHM or mRM specific

This module contains sending and receiving subroutines for data that are specific for mHM or mRM

Authors

Maren Kaluza

Date

Jul 2019

17.9.2 Function/Subroutine Documentation

17.9.2.1 distribute_parameterset()

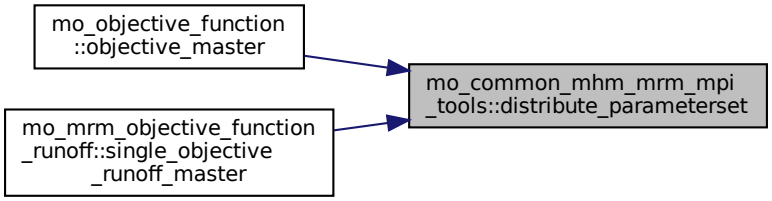
```
subroutine, public mo_common_mhm_mrm_mpi_tools::distribute_parameterset (
            real(dp), dimension(:), intent(in) parameterset )
```

Definition at line 30 of file mo_common_mHM_mRM_MPI_tools.f90.

References mo_common_variables::domainmeta.

Referenced by mo_objective_function::objective_master(), and mo_mrm_objective_function_runoff::single_↩ objective_runoff_master().

Here is the caller graph for this function:



17.9.2.2 get_parameterset()

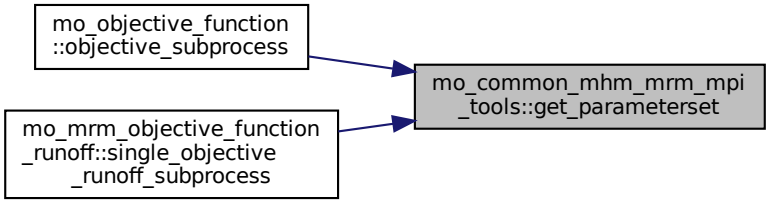
```
subroutine, public mo_common_mhm_mrm_mpi_tools::get_parameterset (
            real(dp), dimension(:), intent(inout), allocatable parameterset )
```

Definition at line 48 of file mo_common_mHM_mRM_MPI_tools.f90.

References mo_common_variables::domainmeta.

Referenced by mo_objective_function::objective_subprocess(), and mo_mrm_objective_function_runoff::single_↩ objective_runoff_subprocess().

Here is the caller graph for this function:



17.10 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.

- subroutine period_copy_period_data (toPeriod, fromPeriod)

17.10.1 Detailed Description

Reading of main model configurations.

This routine reads the configurations of common program parts

Authors

Matthias Zink

Date

Dec 2012

17.10.2 Function/Subroutine Documentation

17.10.2.1 check_optimization_settings()

`subroutine, public mo_common_mhm_mrm_read_config::check_optimization_settings`

TODO: add description.

TODO: add description

Authors

Robert Schweppe

Date

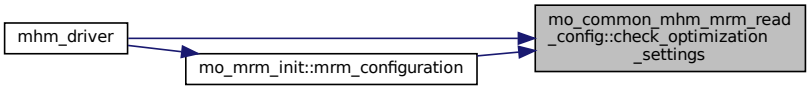
Jun 2018

Definition at line 233 of file mo_common_mHM_mRM_read_config.f90.

References mo_common_mhm_mrm_variables::dds_r, mo_common_variables::global_parameters, 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_configuration().

Here is the caller graph for this function:



17.10.2.2 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

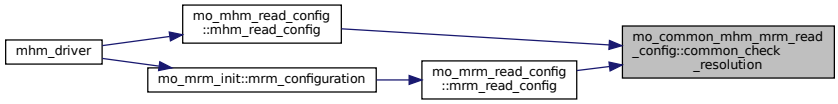
Jun 2018

Definition at line 294 of file mo_common_mHM_mRM_read_config.f90.

References mo_common_variables::domainmeta, 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 caller graph for this function:



17.10.2.3 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

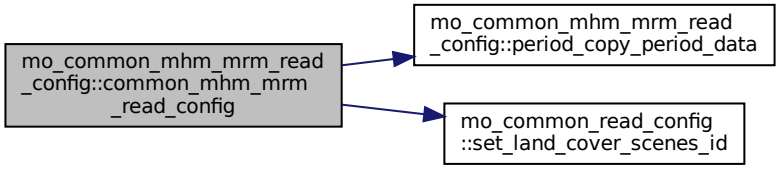
Dec 2012

Definition at line 49 of file mo_common_mHM_mRM_read_config.f90.

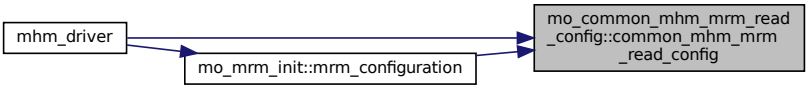
References mo_common_mhm_mrm_variables::dds_r, mo_common_variables::domainmeta, mo_common_mhm_mrm_variables::evalper, mo_common_variables::lcfilename, mo_common_mhm_mrm_variables::lcyearid, mo_common_constants::maxnodomains, mo_common_mhm_mrm_variables::mcmc_error_params, mo_common_mhm_mrm_variables::mcmc_opti, mo_common_mhm_mrm_variables::mhmfilerestartin, mo_common_mhm_mrm_variables::mrm_read_river_network, mo_common_mhm_mrm_variables::mrmfilerestartin, mo_common_mhm_mrm_variables::niterations, mo_common_mhm_mrm_variables::ntstepday, 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, period_copy_period_data(), mo_common_variables::processmatrix, 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_configuration().

Here is the call graph for this function:



Here is the caller graph for this function:



17.10.2.4 period_copy_period_data()

```
subroutine mo_common_mhm_mrm_read_config::period_copy_period_data (
            type(period), intent(inout) toPeriod,
            type(period), intent(in) fromPeriod )
```

Definition at line 354 of file mo_common_mHM_mRM_read_config.f90.

Referenced by common_mhm_mrm_read_config().

Here is the caller graph for this function:

17.11 mo_common_mhm_mrm_restart Module Reference

TODO: add description.

Data Types

- interface check_consistency_element

Functions/Subroutines

- subroutine, public check_dimension_consistency (iBasin, nSoilHorizons_temp, soilHorizonBoundaries_temp, nLAIs_temp, LAIBoundaries_temp, nLandCoverPeriods_temp, landCoverPeriodBoundaries_temp)

  checks dimension configurations read from restart file
- subroutine check_consistency_element_dp (item1, item2, name, iBasin)
- subroutine check_consistency_element_i4 (item1, item2, name, iBasin)

17.11.1 Detailed Description

TODO: add description.

TODO: add description

Authors

Robert Schweppe

Date

Aug 2019

17.11.2 Function/Subroutine Documentation

17.11.2.1 check_consistency_element_dp()

```
subroutine mo_common_mhm_mrm_restart::check_consistency_element_dp (
             real(dp), intent(in) item1,
             real(dp), intent(in) item2,
             character(*), intent(in) name,
             integer(i4), intent(in) iBasin )
```

Definition at line 90 of file mo_common_mHM_mRM_restart.f90.

17.11.2.2 check_consistency_element_i4()

```
subroutine mo_common_mhm_mrm_restart::check_consistency_element_i4 (
             integer(i4), intent(in) item1,
             integer(i4), intent(in) item2,
             character(*), intent(in) name,
             integer(i4), intent(in) iBasin )
```

Definition at line 109 of file mo_common_mHM_mRM_restart.f90.

17.11.2.3 check_dimension_consistency()

```
subroutine, public mo_common_mhm_mrm_restart::check_dimension_consistency (
             integer(i4), intent(in) iBasin,
             integer(i4), intent(in) nSoilHorizons_temp,
             real(dp), dimension(:), intent(inout) soilHorizonBoundaries_temp,
             integer(i4), intent(in) nLAIs_temp,
             real(dp), dimension(:), intent(inout) LAIBoundaries_temp,
             integer(i4), intent(in) nLandCoverPeriods_temp,
             real(dp), dimension(:), intent(inout) landCoverPeriodBoundaries_temp )
```

checks dimension configurations read from restart file

Authors

Robert Schweppe

Date

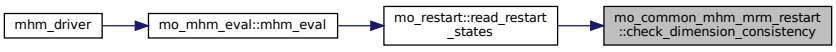
Aug 2019

Definition at line 46 of file mo_common_mHM_mRM_restart.f90.

References mo_mpr_global_variables::horizondepth_mhm, mo_mpr_global_variables::laiboundaries, mo_↩common_variables::lc_year_end, mo_common_variables::lc_year_start, mo_mpr_global_variables::nlai, mo_↩common_variables::nlcoverscene, and mo_mpr_global_variables::nsoilhorizons_mhm.

Referenced by mo_restart::read_restart_states().

Here is the caller graph for this function:



17.12 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
- logical, public mrm_read_river_network
- 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 mhmfilerestartin
- character(256), dimension(:), allocatable, public mrmfilerestartin
- 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

17.12.1 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

17.12.2 Variable Documentation

17.12.2.1 c2tstu

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

Definition at line 32 of file mo_common_mHM_mRM_variables.f90.

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

17.12.2.2 dds_r

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

Definition at line 75 of file mo_common_mHM_mRM_variables.f90.

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().

17.12.2.3 evalper

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

Definition at line 38 of file mo_common_mHM_mRM_variables.f90.

Referenced by mo_common_mhm_mrm_read_config::common_mhm_mrm_read_config(), mo_mrm_river_head↩ ::create_output(), mo_write_fluxes_states::createoutputfile(), mo_mrm_write_fluxes_states::createoutputfile(), 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::readoptidataobs(), mo_write_ascii::write_configfile(), mo_mrm_write::write_configfile(), and mo_mrm_write↩ ::write_daily_obs_sim_discharge().

17.12.2.4 lcyearid

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

Definition at line 42 of file mo_common_mHM_mRM_variables.f90.

Referenced by mo_common_mhm_mrm_read_config::common_mhm_mrm_read_config(), mo_common_↩ datetime_type::datetimeinfo_init(), mo_mhm_eval::mhm_eval(), mo_write_ascii::write_configfile(), and mo_mrm_↩ write::write_configfile().

17.12.2.5 mcmc_error_params

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

Definition at line 89 of file mo_common_mHM_mRM_variables.f90.

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

17.12.2.6 mcmc_opti

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

Definition at line 85 of file mo_common_mHM_mRM_variables.f90.

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

17.12.2.7 mhmfilerestartin

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

Definition at line 53 of file mo_common_mHM_mRM_variables.f90.

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

17.12.2.8 mrm_coupling_mode

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

Definition at line 26 of file mo_common_mHM_mRM_variables.f90.

Referenced by mo_mrm_init::mrm_configuration(), mo_mrm_init::mrm_init(), mo_mrm_write::mrm_write(), and mo_mrm_write::write_configfile().

17.12.2.9 mrm_read_river_network

```
logical, public mo_common_mhm_mrm_variables::mrm_read_river_network
```

Definition at line 35 of file mo_common_mHM_mRM_variables.f90.

Referenced by mo_common_mhm_mrm_read_config::common_mhm_mrm_read_config(), and mo_mrm_init↩::mrm_init().

17.12.2.10 mrmfilerestartin

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

Definition at line 54 of file mo_common_mHM_mRM_variables.f90.

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

17.12.2.11 nerror_model

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

Definition at line 87 of file mo_common_mHM_mRM_variables.f90.

17.12.2.12 niterations

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

Definition at line 74 of file mo_common_mHM_mRM_variables.f90.

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().

17.12.2.13 ntstepday

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

Definition at line 47 of file mo_common_mHM_mRM_variables.f90.

Referenced by mo_meteo_forcings::chunk_size(), mo_common_mhm_mrm_read_config::common_mhm_mrm_read_config(), mo_common_datetime_type::datetimeinfo_init(), 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_read_data::mrm_read_discharge(), and mo_mrm_write::mrm_write().

17.12.2.14 opti_function

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

Definition at line 63 of file mo_common_mHM_mRM_variables.f90.

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_objective_function::objective_master(), mo_objective_function::objective_subprocess(), mo_optimization::optimization(), mo_mrm_objective_function_runoff::single_objective_runoff(), mo_mrm_objective_function_runoff::single_objective_runoff_master(), and mo_mrm_objective_function_runoff::single_objective_runoff_subprocess().

17.12.2.15 opti_method

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

Definition at line 59 of file mo_common_mHM_mRM_variables.f90.

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

17.12.2.16 optimize

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

Definition at line 67 of file mo_common_mHM_mRM_variables.f90.

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(), mo_mrm_mpr::mrm_init_param(), mo_mrm_read_data::mrm_read_discharge(), and mo_mrm_mpr::mrm_update_param().

17.12.2.17 optimize_restart

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

Definition at line 69 of file mo_common_mHM_mRM_variables.f90.

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

17.12.2.18 read_restart

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

Definition at line 34 of file mo_common_mHM_mRM_variables.f90.

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

17.12.2.19 readper

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

Definition at line 40 of file mo_common_mHM_mRM_variables.f90.

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

17.12.2.20 resolutionrouting

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

Definition at line 33 of file mo_common_mHM_mRM_variables.f90.

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_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().

17.12.2.21 sa_temp

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

Definition at line 77 of file mo_common_mHM_mRM_variables.f90.

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

17.12.2.22 sce_ngs

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

Definition at line 79 of file mo_common_mHM_mRM_variables.f90.

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().

17.12.2.23 sce_npg

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

Definition at line 81 of file mo_common_mHM_mRM_variables.f90.

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().

17.12.2.24 sce_nps

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

Definition at line 83 of file mo_common_mHM_mRM_variables.f90.

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().

17.12.2.25 seed

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

Definition at line 72 of file mo_common_mHM_mRM_variables.f90.

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

17.12.2.26 simper

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

Definition at line 39 of file mo_common_mHM_mRM_variables.f90.

Referenced by mo_meteo_forcings::chunk_size(), mo_common_mhm_mrm_read_config::common_mhm_mrm_↩ read_config(), mo_common_datetime_type::datetimeinfo_init(), mo_write_fluxes_states::fluxesunit(), mo_meteo↩ _forcings::is_read(), mo_mhm_eval::mhm_eval(), mo_mrm_read_data::mrm_read_discharge(), mo_mrm_read_↩ data::mrm_read_total_runoff(), mo_mrm_write::mrm_write(), mo_write_ascii::write_configfile(), and mo_mrm_↩ write::write_configfile().

17.12.2.27 timestep

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

Definition at line 31 of file mo_common_mHM_mRM_variables.f90.

Referenced by mo_common_mhm_mrm_read_config::common_mhm_mrm_read_config(), mo_startup↩ ::constants_init(), mo_common_datetime_type::datetimeinfo_increment(), mo_common_datetime_type::datetimeinfo↩ _init(), mo_write_fluxes_states::fluxesunit(), mo_meteo_forcings::is_read(), mo_mhm_eval::mhm_eval(), mo↩ _mrm_mpr::mrm_init_param(), 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().

17.12.2.28 warmingdays

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

Definition at line 41 of file mo_common_mHM_mRM_variables.f90.

Referenced by mo_common_mhm_mrm_read_config::common_mhm_mrm_read_config(), mo_mrm_objective_↩ function_runoff::extract_runoff(), mo_mhm_eval::mhm_eval(), and mo_mrm_write::mrm_write().

17.12.2.29 warmper

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

Definition at line 37 of file mo_common_mHM_mRM_variables.f90.

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

17.13 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)

  Read main configurations commonly used by mHM, mRM and MPR.
- subroutine init_domain_variable (nDomains, optiData, domainMeta)
- subroutine init_domain_variable_for_master (domainMeta, colMasters, colDomain)
- subroutine distributedomainsroundrobin (nproc, rank, domainMeta)
- subroutine distribute_processes_to_domains_according_to_role (optiData, rank, domainMeta, colMasters, colDomain)

17.13.1 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

17.13.2 Function/Subroutine Documentation

17.13.2.1 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

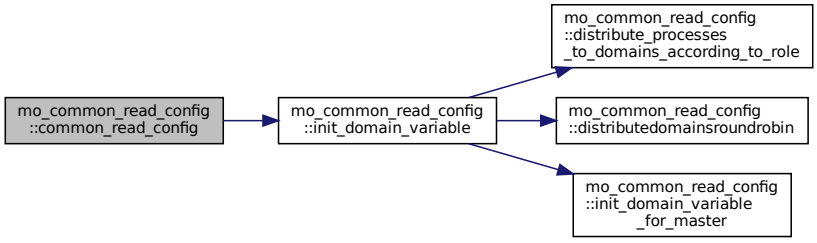
Dec 2012

Definition at line 50 of file mo_common_read_config.f90.

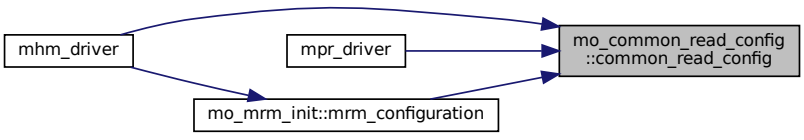
References mo_common_variables::contact, mo_common_variables::conventions, mo_common_variables↩ ::dircommonfiles, mo_common_variables::dirconfigout, mo_common_variables::dirlcover, mo_common_variables↩ ::dirmorpho, mo_common_variables::dirout, mo_common_variables::domainmeta, mo_common_variables↩ ::filelatlon, mo_common_variables::history, mo_common_variables::iflag_cordinate_sys, init_domain_variable(), mo_common_variables::lc_year_end, mo_common_variables::lc_year_start, mo_common_variables::lcfilename, mo_common_constants::maxnlcovers, mo_common_constants::maxnodomains, mo_common_variables::mhm↩ _details, mo_common_variables::mhmfilerestartout, mo_common_variables::mrmfilerestartout, mo_common↩ _variables::nlcoverscene, mo_common_variables::nprocesses, mo_common_variables::nuniquel0domains, 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_configuration().

Here is the call graph for this function:



Here is the caller graph for this function:



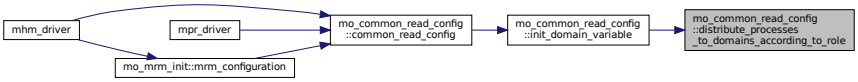
17.13.2.2 distribute_processes_to_domains_according_to_role()

```
subroutine mo_common_read_config::distribute_processes_to_domains_according_to_role (
             integer(i4), dimension(:), intent(in) optiData,
             integer(i4), intent(in) rank,
             type(domain_meta), intent(inout) domainMeta,
             integer(i4), intent(out) colMasters,
             integer(i4), intent(out) colDomain )
```

Definition at line 478 of file mo_common_read_config.f90.

Referenced by init_domain_variable().

Here is the caller graph for this function:



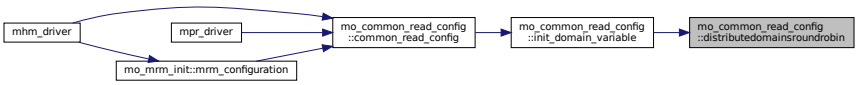
17.13.2.3 distributedomainsroundrobin()

```
subroutine mo_common_read_config::distributedomainsroundrobin (
             integer(i4), intent(in) nproc,
             integer(i4), intent(in) rank,
             type(domain_meta), intent(inout) domainMeta )
```

Definition at line 455 of file mo_common_read_config.f90.

Referenced by init_domain_variable().

Here is the caller graph for this function:



17.13.2.4 init_domain_variable()

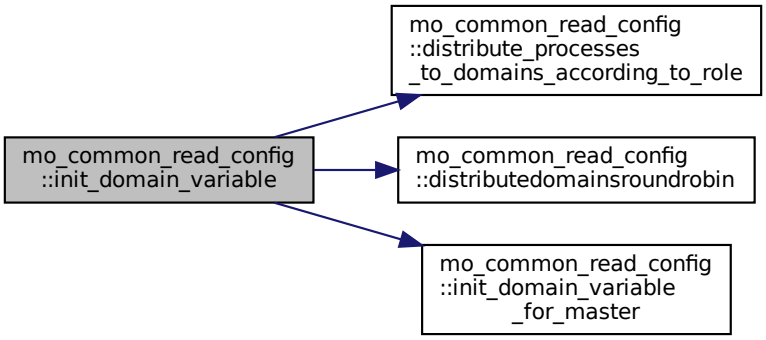
```
subroutine mo_common_read_config::init_domain_variable (
             integer(i4), intent(in) nDomains,
             integer(i4), dimension(:), intent(in) optiData,
             type(domain_meta), intent(inout) domainMeta )
```

Definition at line 358 of file mo_common_read_config.f90.

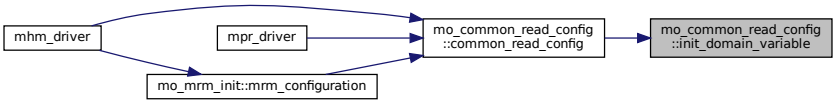
References mo_common_variables::comm, distribute_processes_to_domains_according_to_role(), distributedomainsroundrobin(), and init_domain_variable_for_master().

Referenced by common_read_config().

Here is the call graph for this function:



Here is the caller graph for this function:



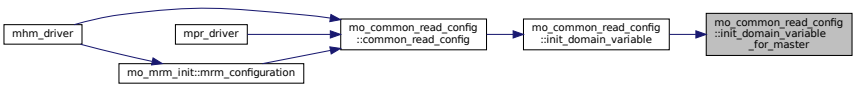
17.13.2.5 init_domain_variable_for_master()

```
subroutine mo_common_read_config::init_domain_variable_for_master (
             type(domain_meta), intent(inout) domainMeta,
             integer(i4), intent(out) colMasters,
             integer(i4), intent(out) colDomain )
```

Definition at line 435 of file mo_common_read_config.f90.

Referenced by init_domain_variable().

Here is the caller graph for this function:



17.13.2.6 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 )
```

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

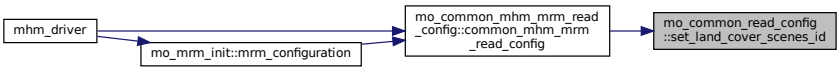
Dec 2012

Definition at line 270 of file mo_common_read_config.f90.

References mo_common_variables::domainmeta, mo_common_variables::lc_year_end, mo_common_variables↩ ::lc_year_start, 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 caller graph for this function:



17.14 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.

17.14.1 Detailed Description

TODO: add description.

TODO: add description

Authors

Robert Schweppe

Date

Jun 2018

17.14.2 Function/Subroutine Documentation

17.14.2.1 read_dem()

```
subroutine, public mo_common_read_data::read_dem
```

TODO: add description.

TODO: add description

Authors

Robert Schweppe

Date

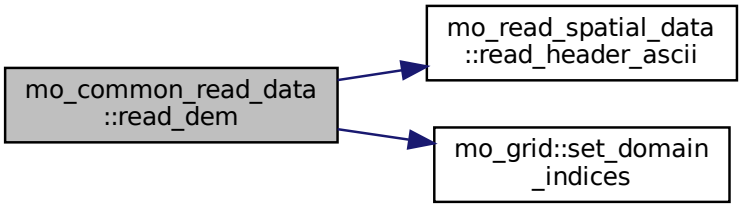
Jun 2018

Definition at line 41 of file mo_common_read_data.f90.

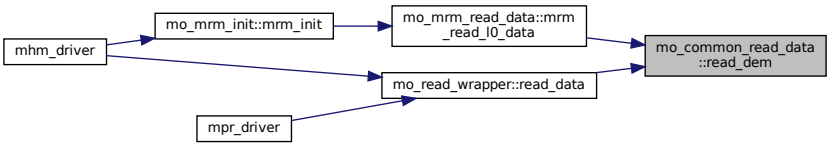
References mo_common_variables::dirmorpho, mo_common_variables::domainmeta, mo_common_file::file_dem, mo_common_variables::l0_elev, mo_common_variables::level0, mo_common_constants::nodata_dp, mo_read↩_spatial_data::read_header_ascii(), mo_common_variables::resolutionhydrology, mo_grid::set_domain_indices(), 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:



17.14.2.2 read_lcover()

```
subroutine, public mo_common_read_data::read_lcover
```

TODO: add description.

TODO: add description

Authors

Robert Schweppe

Date

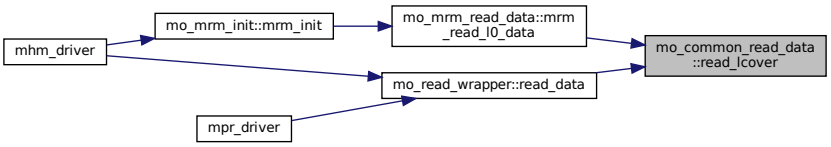
Jun 2018

Definition at line 143 of file mo_common_read_data.f90.

References mo_common_variables::dirlcover, mo_common_variables::domainmeta, mo_common_variables::l0_↩
lcover, mo_common_variables::lcfilename, mo_common_variables::level0, 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 caller graph for this function:



17.15 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 domain

- subroutine, public read_grid_info (domainID, InFile, level_name, new_grid)

  reads configuration apart from Level 11 configuration from a restart directory

17.15.1 Detailed Description

TODO: add description.

TODO: add description

Authors

Robert Schweppe

Date

Jun 2018

17.15.2 Function/Subroutine Documentation

17.15.2.1 read_grid_info()

```
subroutine, public mo_common_restart::read_grid_info (
             integer(i4), intent(in) domainID,
             character(256), intent(in) InFile,
             character(*), intent(in) level_name,
             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) :: iDomain*	number of domain
in	*character(256) :: InFile*	Input Path including trailing slash
in	*character(∗) :: level_name*	level_name (id)
in,out	*type(Grid) :: new_grid*	grid to save information to

Authors

Stephan Thober

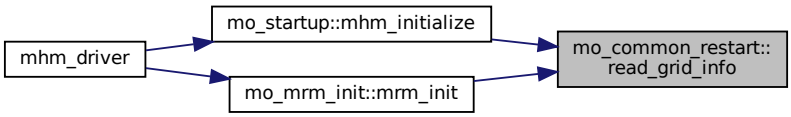
Date

Apr 2013

Definition at line 162 of file mo_common_restart.f90.

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

Here is the caller graph for this function:



17.15.2.2 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 domain

write restart files for each domain. For each domain three restart files are written. These are xxx_states.nc, xxx_↩ L11_config.nc, and xxx_config.nc (xxx being the three digit domain 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

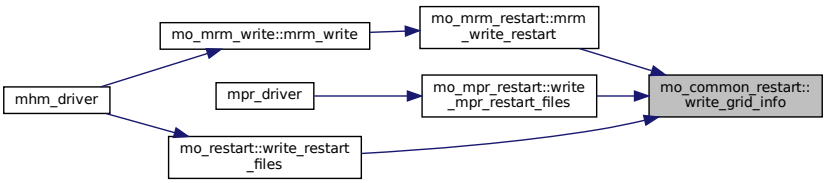
Jun 2014

Definition at line 62 of file mo_common_restart.f90.

References 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:



17.16 mo_common_variables Module Reference

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

Data Types

- type period
- type grid
- type gridremapper
- type domain_meta

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_domain
- logical, public write_restart
- character(256), dimension(:), allocatable, public mhmfilerestartout
- character(256), dimension(:), allocatable, public mrmfilerestartout
- 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
- real(dp), dimension(:), allocatable, public l0_elev
- integer(i4), dimension(:, :), allocatable, public l0_lcover
- type(mpi_comm) comm
- type(domain_meta), public domainmeta
- integer(i4), public nuniquel0domains
- 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 = 11
- 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

17.16.1 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

17.16.2 Variable Documentation

17.16.2.1 alma_convention

```
logical mo_common_variables::alma_convention
```

Definition at line 209 of file mo_common_variables.f90.

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

17.16.2.2 comm

```
type(mpi_comm) mo_common_variables::comm
```

Definition at line 131 of file mo_common_variables.f90.

Referenced by mo_common_read_config::init_domain_variable().

17.16.2.3 contact

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

Definition at line 33 of file mo_common_variables.f90.

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

17.16.2.4 conventions

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

Definition at line 32 of file mo_common_variables.f90.

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

17.16.2.5 dircommonfiles

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

Definition at line 52 of file mo_common_variables.f90.

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

17.16.2.6 dirconfigout

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

Definition at line 51 of file mo_common_variables.f90.

Referenced by mo_common_read_config::common_read_config(), 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().

17.16.2.7 dirlcover

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

Definition at line 54 of file mo_common_variables.f90.

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().

17.16.2.8 dirmorpho

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

Definition at line 53 of file mo_common_variables.f90.

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().

17.16.2.9 dirout

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

Definition at line 55 of file mo_common_variables.f90.

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

17.16.2.10 domainmeta

```
type(domain_meta), public mo_common_variables::domainmeta
```

Definition at line 161 of file mo_common_variables.f90.

Referenced by mo_mrm_river_head::calc_channel_elevation(), mo_mrm_net_startup::celllength(), 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_riv_temp_class::config(), mo_mrm_init::config_output(), mo_common_mhm_mrm_mpi_tools::distribute_parameterset(), mo_common_mhm_mrm_mpi_tools::get_parameterset(), 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_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_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_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_rmse_tws(), mo_objective_function::objective_kge_q_sm_corr(), mo_objective_function::objective_master(), mo_objective_function::objective_neutrons_kge_catchment_avg(), mo_objective_function::objective_q_et_tws_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_objective_function::objective_subprocess(), mo_optimization::optimization(), mo_meteo_forcings::prepare_meteo_forcings_data(), 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_mrm_objective_function_runoff::single_objective_runoff_master(), mo_mrm_objective_function_runoff::single_objective_runoff_subprocess(), mo_mrm_init::variables_alloc_routing(), 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(), and mo_restart::write_restart_files().

17.16.2.11 filelatlon

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

Definition at line 56 of file mo_common_variables.f90.

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

17.16.2.12 global_parameters

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

Definition at line 197 of file mo_common_variables.f90.

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(), 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().

17.16.2.13 global_parameters_name

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

Definition at line 201 of file mo_common_variables.f90.

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

17.16.2.14 history

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

Definition at line 35 of file mo_common_variables.f90.

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

17.16.2.15 iflag_cordinate_sys

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

Definition at line 40 of file mo_common_variables.f90.

Referenced by mo_mrm_river_head::calc_slope(), mo_common_read_config::common_read_config(), mo_↩ write_fluxes_states::createoutputfile(), mo_mrm_write_fluxes_states::createoutputfile(), mo_grid::l0_grid_setup(), mo_mrm_net_startup::l11_stream_features(), mo_mrm_mpr::mrm_init_param(), mo_mrm_mpr::mrm_update↩ _param(), mo_mrm_write_fluxes_states::newoutputdataset(), mo_write_fluxes_states::newoutputdataset(), and mo_write_ascii::write_configfile().

17.16.2.16 l0_domain

```
integer(i4), dimension(:), allocatable, public mo_common_variables::l0_domain
```

Definition at line 42 of file mo_common_variables.f90.

17.16.2.17 l0_elev

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

Definition at line 122 of file mo_common_variables.f90.

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().

17.16.2.18 l0_l1_remap

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

Definition at line 115 of file mo_common_variables.f90.

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

17.16.2.19 l0_lcover

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

Definition at line 124 of file mo_common_variables.f90.

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().

17.16.2.20 lc_year_end

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

Definition at line 171 of file mo_common_variables.f90.

Referenced by mo_common_mhm_mrm_restart::check_dimension_consistency(), mo_common_read_config↩ ::common_read_config(), mo_mrm_restart::mrm_write_restart(), 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_eff_params(), mo_mpr_restart::write_mpr_restart_files(), and mo_restart::write_restart_↩ files().

17.16.2.21 lc_year_start

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

Definition at line 170 of file mo_common_variables.f90.

Referenced by mo_common_mhm_mrm_restart::check_dimension_consistency(), mo_common_read_config↩ ::common_read_config(), mo_mrm_restart::mrm_write_restart(), 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_eff_params(), mo_mpr_restart::write_mpr_restart_files(), and mo_restart::write_restart_↩ files().

17.16.2.22 lcfilename

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

Definition at line 169 of file mo_common_variables.f90.

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().

17.16.2.23 level0

```
type(grid), dimension(:), allocatable, target, public mo_common_variables::level0
```

Definition at line 98 of file mo_common_variables.f90.

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().

17.16.2.24 level1

```
type(grid), dimension(:), allocatable, target, public mo_common_variables::level1
```

Definition at line 99 of file mo_common_variables.f90.

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_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_kge_q_et(), 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(), mo_restart::read_restart_states(), mo_read_optional_data::readoptidataobs(), mo_write_ascii::write_configfile(), mo_mrm_write::write_configfile(), mo_mpr_restart::write_mpr_restart_files(), and mo_restart::write_restart_files().

17.16.2.25 mhm_details

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

Definition at line 34 of file mo_common_variables.f90.

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

17.16.2.26 mhmfilerestartout

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

Definition at line 49 of file mo_common_variables.f90.

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

17.16.2.27 mrmfilerestartout

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

Definition at line 50 of file mo_common_variables.f90.

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

17.16.2.28 nlcoverscene

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

Definition at line 168 of file mo_common_variables.f90.

Referenced by mo_common_mhm_mrm_restart::check_dimension_consistency(), 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().

17.16.2.29 nprocesses

```
integer(i4), parameter, public mo_common_variables::nprocesses = 11
```

Definition at line 176 of file mo_common_variables.f90.

Referenced by mo_common_read_config::common_read_config(), and mo_write_ascii::write_optinamelist().

17.16.2.30 nuniquel0domains

```
integer(i4), public mo_common_variables::nuniquel0domains
```

Definition at line 162 of file mo_common_variables.f90.

Referenced by mo_common_read_config::common_read_config().

17.16.2.31 processmatrix

```
integer(i4), dimension(nprocesses, 3), public mo_common_variables::processmatrix
```

Definition at line 188 of file mo_common_variables.f90.

Referenced by mo_common_mhm_mrm_read_config::common_mhm_mrm_read_config(), 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_init::mrm_configuration(), 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().

17.16.2.32 project_details

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

Definition at line 29 of file mo_common_variables.f90.

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

17.16.2.33 resolutionhydrology

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

Definition at line 41 of file mo_common_variables.f90.

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_init::mrm_init(), mo_common_read_data::read_dem(), mo_write_ascii::write_configfile(), and mo_mrm_write::write_configfile().

17.16.2.34 setup_description

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

Definition at line 30 of file mo_common_variables.f90.

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

17.16.2.35 simulation_type

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

Definition at line 31 of file mo_common_variables.f90.

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

17.16.2.36 write_restart

```
logical, public mo_common_variables::write_restart
```

Definition at line 43 of file mo_common_variables.f90.

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().

17.17 mo_file Module Reference

Provides file names and units for mHM.

Variables

- character(len=∗), parameter version = '5.11.2'

  Current mHM model version (will be set to 5.11.2)
- character(len=∗), parameter version_date = 'Jul 2021'

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.

17.17.1 Detailed Description

Provides file names and units for mHM.

Provides all filenames as well as all units used for the hydrologic model mHM. The `version` parameter will be set during compilation to " 5.11.2". The `version_date` parameter will be set during compilation to " Jul 2021", if it is a release version, otherwise it will be the current date.

Authors

Matthias Cuntz

Sebastian Mueller

Date

Jan 2012

17.17.2 Variable Documentation

17.17.2.1 file_defoutput

```
character(len = *), parameter mo_file::file_defoutput = 'mhm_outputs.nml'
```

file defining mHM's outputs

Definition at line 48 of file mo_file.f90.

Referenced by mo_mhm_read_config::mhm_read_config().

17.17.2.2 file_main

```
character(len = *), parameter mo_file::file_main = 'mhm_driver.f90'
```

Driver file.

Definition at line 38 of file mo_file.f90.

Referenced by mhm_driver().

17.17.2.3 file_namelist_mhm

```
character(len = *), parameter mo_file::file_namelist_mhm = 'mhm.nml'
```

Namelist file name.

Definition at line 40 of file mo_file.f90.

17.17.2.4 file_namelist_mhm_param

```
character(len = *), parameter mo_file::file_namelist_mhm_param = 'mhm_parameter.nml'
```

Parameter namelists file name.

Definition at line 44 of file mo_file.f90.

Referenced by mo_startup::constants_init().

17.17.2.5 udefoutput

```
integer, parameter mo_file::udefoutput = 67
```

Unit for file defining mHM's outputs.

Definition at line 50 of file mo_file.f90.

Referenced by mo_mhm_read_config::mhm_read_config().

17.17.2.6 unamelist_mhm

```
integer, parameter mo_file::unamelist_mhm = 30
```

Unit for namelist.

Definition at line 42 of file mo_file.f90.

17.17.2.7 unamelist_mhm_param

```
integer, parameter mo_file::unamelist_mhm_param = 31
```

Unit for namelist.

Definition at line 46 of file mo_file.f90.

17.17.2.8 version

```
character(len = *), parameter mo_file::version = '5.11.2'
```

Current mHM model version (will be set to 5.11.2)

Definition at line 31 of file mo_file.f90.

Referenced by mo_mrm_river_head::create_output(), mo_write_fluxes_states::createoutputfile(), mhm_driver(), and mo_write_ascii::write_configfile().

17.17.2.9 version_date

```
character(len = *), parameter mo_file::version_date = 'Jul 2021'
```

Time of current mHM model version release.

Definition at line 34 of file mo_file.f90.

Referenced by mhm_driver().

17.18 mo_global_variables Module Reference

Global variables ONLY used in reading, writing and startup.

Variables

- integer(i4) output_deflate_level
- logical output_double_precision
- 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
- 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 dirradiation
- integer(i4), parameter, public routingstates = 2
- 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_ssrd
- real(dp), dimension(:, :), allocatable, public l1_strd
- real(dp), dimension(:, :), allocatable, public l1_tann
- real(dp), dimension(:, :), allocatable, public l1_sm
- logical, dimension(:, :), allocatable, public l1_sm_mask
- real(dp), dimension(:, :), allocatable, public l1_neutronsdata
- logical, dimension(:, :), allocatable, public l1_neutronsdata_mask
- integer(i4) nsoilhorizons_sm_input
- type(optidata), dimension(:), allocatable, public l1_smobs
- type(optidata), dimension(:), allocatable, public l1_neutronsobs

- type(optidata), dimension(:), allocatable, public l1_etobs
- type(optidata), dimension(:), allocatable, public l1_twsaobs
- 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(yearmonths_i4), public evap_coeff
- real(dp), dimension(yearmonths_i4), public fday_prec
- real(dp), dimension(yearmonths_i4), public fnight_prec
- real(dp), dimension(yearmonths_i4), public fday_pet
- real(dp), dimension(yearmonths_i4), public fnight_pet
- real(dp), dimension(yearmonths_i4), public fday_temp
- real(dp), dimension(yearmonths_i4), public fnight_temp
- real(dp), dimension(yearmonths_i4), public fday_ssrd
- real(dp), dimension(yearmonths_i4), public fnight_ssrd
- real(dp), dimension(yearmonths_i4), public fday_strd
- real(dp), dimension(yearmonths_i4), public fnight_strd
- real(dp), dimension(:), allocatable, public neutron_integral_afast

17.18.1 Detailed Description

Global variables ONLY used in reading, writing and startup.

TODO: add description

Authors

Luis Samaniego

Date

Dec 2012

17.18.2 Variable Documentation

17.18.2.1 dirabsvappressure

```
character(256), dimension(:), allocatable, public mo_global_variables::dirabsvappressure
```

Definition at line 76 of file mo_global_variables.f90.

Referenced by mo_mhm_read_config::mhm_read_config(), and mo_meteo_forcings::prepare_meteo_forcings_↩data().

17.18.2.2 dirmaxtemperature

```
character(256), dimension(:), allocatable, public mo_global_variables::dirmaxtemperature
```

Definition at line 74 of file mo_global_variables.f90.

Referenced by mo_mhm_read_config::mhm_read_config(), and mo_meteo_forcings::prepare_meteo_forcings_↩data().

17.18.2.3 dirmintemperature

```
character(256), dimension(:), allocatable, public mo_global_variables::dirmintemperature
```

Definition at line 73 of file mo_global_variables.f90.

Referenced by mo_mhm_read_config::mhm_read_config(), and mo_meteo_forcings::prepare_meteo_forcings_↩data().

17.18.2.4 dirnetradiation

```
character(256), dimension(:), allocatable, public mo_global_variables::dirnetradiation
```

Definition at line 75 of file mo_global_variables.f90.

Referenced by mo_mhm_read_config::mhm_read_config(), and mo_meteo_forcings::prepare_meteo_forcings_↩data().

17.18.2.5 dirprecipitation

```
character(256), dimension(:), allocatable, public mo_global_variables::dirprecipitation
```

Definition at line 71 of file mo_global_variables.f90.

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().

17.18.2.6 dirradiation

```
character(256), dimension(:), allocatable, public mo_global_variables::dirradiation
```

Definition at line 80 of file mo_global_variables.f90.

Referenced by mo_mhm_read_config::mhm_read_config(), and mo_meteo_forcings::prepare_meteo_forcings_↩data().

17.18.2.7 dirreferenceet

```
character(256), dimension(:), allocatable, public mo_global_variables::dirreferenceet
```

Definition at line 78 of file mo_global_variables.f90.

Referenced by mo_mhm_read_config::mhm_read_config(), mo_meteo_forcings::prepare_meteo_forcings_data(), and mo_write_ascii::write_configfile().

17.18.2.8 dirtemperature

```
character(256), dimension(:), allocatable, public mo_global_variables::dirtemperature
```

Definition at line 72 of file mo_global_variables.f90.

Referenced by mo_mhm_read_config::mhm_read_config(), mo_meteo_forcings::prepare_meteo_forcings_data(), and mo_write_ascii::write_configfile().

17.18.2.9 dirwindspeed

```
character(256), dimension(:), allocatable, public mo_global_variables::dirwindspeed
```

Definition at line 77 of file mo_global_variables.f90.

Referenced by mo_mhm_read_config::mhm_read_config(), and mo_meteo_forcings::prepare_meteo_forcings_↩ data().

17.18.2.10 evap_coeff

```
real(dp), dimension(yearmonths_i4), public mo_global_variables::evap_coeff
```

Definition at line 173 of file mo_global_variables.f90.

Referenced by mo_mhm_eval::mhm_eval(), and mo_mhm_read_config::mhm_read_config().

17.18.2.11 fday_pet

```
real(dp), dimension(yearmonths_i4), public mo_global_variables::fday_pet
```

Definition at line 176 of file mo_global_variables.f90.

Referenced by mo_mhm_eval::mhm_eval(), and mo_mhm_read_config::mhm_read_config().

17.18.2.12 fday_prec

```
real(dp), dimension(yearmonths_i4), public mo_global_variables::fday_prec
```

Definition at line 174 of file mo_global_variables.f90.

Referenced by mo_mhm_eval::mhm_eval(), and mo_mhm_read_config::mhm_read_config().

17.18.2.13 fday_ssrd

```
real(dp), dimension(yearmonths_i4), public mo_global_variables::fday_ssrd
```

Definition at line 180 of file mo_global_variables.f90.

Referenced by mo_mhm_eval::mhm_eval(), and mo_mhm_read_config::mhm_read_config().

17.18.2.14 fday_strd

```
real(dp), dimension(yearmonths_i4), public mo_global_variables::fday_strd
```

Definition at line 182 of file mo_global_variables.f90.

Referenced by mo_mhm_eval::mhm_eval(), and mo_mhm_read_config::mhm_read_config().

17.18.2.15 fday_temp

```
real(dp), dimension(yearmonths_i4), public mo_global_variables::fday_temp
```

Definition at line 178 of file mo_global_variables.f90.

Referenced by mo_mhm_eval::mhm_eval(), and mo_mhm_read_config::mhm_read_config().

17.18.2.16 fnight_pet

```
real(dp), dimension(yearmonths_i4), public mo_global_variables::fnight_pet
```

Definition at line 177 of file mo_global_variables.f90.

Referenced by mo_mhm_eval::mhm_eval(), and mo_mhm_read_config::mhm_read_config().

17.18.2.17 fnight_prec

```
real(dp), dimension(yearmonths_i4), public mo_global_variables::fnight_prec
```

Definition at line 175 of file mo_global_variables.f90.

Referenced by mo_mhm_eval::mhm_eval(), and mo_mhm_read_config::mhm_read_config().

17.18.2.18 fnight_ssrd

```
real(dp), dimension(yearmonths_i4), public mo_global_variables::fnight_ssrd
```

Definition at line 181 of file mo_global_variables.f90.

Referenced by mo_mhm_eval::mhm_eval(), and mo_mhm_read_config::mhm_read_config().

17.18.2.19 fnight_strd

```
real(dp), dimension(yearmonths_i4), public mo_global_variables::fnight_strd
```

Definition at line 183 of file mo_global_variables.f90.

Referenced by mo_mhm_eval::mhm_eval(), and mo_mhm_read_config::mhm_read_config().

17.18.2.20 fnight_temp

```
real(dp), dimension(yearmonths_i4), public mo_global_variables::fnight_temp
```

Definition at line 179 of file mo_global_variables.f90.

Referenced by mo_mhm_eval::mhm_eval(), and mo_mhm_read_config::mhm_read_config().

17.18.2.21 inputformat_meteo_forcings

```
character(256), public mo_global_variables::inputformat_meteo_forcings
```

Definition at line 65 of file mo_global_variables.f90.

Referenced by mo_mhm_read_config::mhm_read_config(), and mo_meteo_forcings::prepare_meteo_forcings_↩data().

17.18.2.22 l1_absvappress

```
real(dp), dimension(:, :), allocatable, public mo_global_variables::l1_absvappress
```

Definition at line 107 of file mo_global_variables.f90.

Referenced by mo_mhm_eval::mhm_eval(), and mo_meteo_forcings::prepare_meteo_forcings_data().

17.18.2.23 l1_aetcanopy

```
real(dp), dimension(:), allocatable, public mo_global_variables::l1_aetcanopy
```

Definition at line 153 of file mo_global_variables.f90.

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().

17.18.2.24 l1_aetsealed

```
real(dp), dimension(:), allocatable, public mo_global_variables::l1_aetsealed
```

Definition at line 154 of file mo_global_variables.f90.

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().

17.18.2.25 l1_aetsoil

```
real(dp), dimension(:, :), allocatable, public mo_global_variables::l1_aetsoil
```

Definition at line 152 of file mo_global_variables.f90.

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().

17.18.2.26 l1_baseflow

```
real(dp), dimension(:), allocatable, public mo_global_variables::l1_baseflow
```

Definition at line 155 of file mo_global_variables.f90.

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().

17.18.2.27 l1_etobs

```
type(optidata), dimension(:), allocatable, public mo_global_variables::l1_etobs
```

Definition at line 129 of file mo_global_variables.f90.

Referenced by mo_objective_function::create_domain_avg_et(), mo_mhm_eval::mhm_eval(), mo_mhm_read_↩ config::mhm_read_config(), mo_objective_function::objective_kge_q_et(), mo_objective_function::objective_kge↩ _q_rmse_et(), and mo_objective_function::objective_q_et_tws_kge_catchment_avg().

17.18.2.28 l1_fastrunoff

```
real(dp), dimension(:), allocatable, public mo_global_variables::l1_fastrunoff
```

Definition at line 157 of file mo_global_variables.f90.

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().

17.18.2.29 l1_infilsoil

```
real(dp), dimension(:, :), allocatable, public mo_global_variables::l1_infilsoil
```

Definition at line 156 of file mo_global_variables.f90.

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().

17.18.2.30 l1_inter

```
real(dp), dimension(:), allocatable, public mo_global_variables::l1_inter
```

Definition at line 140 of file mo_global_variables.f90.

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().

17.18.2.31 l1_melt

```
real(dp), dimension(:), allocatable, public mo_global_variables::l1_melt
```

Definition at line 158 of file mo_global_variables.f90.

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().

17.18.2.32 l1_netrad

```
real(dp), dimension(:, :), allocatable, public mo_global_variables::l1_netrad
```

Definition at line 106 of file mo_global_variables.f90.

Referenced by mo_mhm_eval::mhm_eval(), and mo_meteo_forcings::prepare_meteo_forcings_data().

17.18.2.33 l1_neutrons

```
real(dp), dimension(:), allocatable, public mo_global_variables::l1_neutrons
```

Definition at line 146 of file mo_global_variables.f90.

Referenced by mo_mhm_eval::mhm_eval(), mo_init_states::variables_alloc(), and mo_init_states::variables_↩ default_init().

17.18.2.34 l1_neutronsdata

```
real(dp), dimension(:, :), allocatable, public mo_global_variables::l1_neutronsdata
```

Definition at line 119 of file mo_global_variables.f90.

17.18.2.35 l1_neutronsdata_mask

```
logical, dimension(:, :), allocatable, public mo_global_variables::l1_neutronsdata_mask
```

Definition at line 120 of file mo_global_variables.f90.

17.18.2.36 l1_neutronsobs

```
type(optidata), dimension(:), allocatable, public mo_global_variables::l1_neutronsobs
```

Definition at line 127 of file mo_global_variables.f90.

Referenced by mo_mhm_eval::mhm_eval(), mo_mhm_read_config::mhm_read_config(), and mo_objective_↩ function::objective_neutrons_kge_catchment_avg().

17.18.2.37 l1_percol

```
real(dp), dimension(:), allocatable, public mo_global_variables::l1_percol
```

Definition at line 159 of file mo_global_variables.f90.

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().

17.18.2.38 l1_pet

```
real(dp), dimension(:, :), allocatable, public mo_global_variables::l1_pet
```

Definition at line 103 of file mo_global_variables.f90.

Referenced by mo_mhm_eval::mhm_eval(), and mo_meteo_forcings::prepare_meteo_forcings_data().

17.18.2.39 l1_pet_calc

```
real(dp), dimension(:), allocatable, public mo_global_variables::l1_pet_calc
```

Definition at line 151 of file mo_global_variables.f90.

Referenced by mo_mhm_eval::mhm_eval(), mo_init_states::variables_alloc(), and mo_init_states::variables_↩ default_init().

17.18.2.40 l1_pet_weights

```
real(dp), dimension(:, :, :), allocatable, public mo_global_variables::l1_pet_weights
```

Definition at line 99 of file mo_global_variables.f90.

Referenced by mo_mhm_eval::mhm_eval(), and mo_meteo_forcings::prepare_meteo_forcings_data().

17.18.2.41 l1_pre

```
real(dp), dimension(:, :), allocatable, public mo_global_variables::l1_pre
```

Definition at line 101 of file mo_global_variables.f90.

Referenced by mo_mhm_eval::mhm_eval(), and mo_meteo_forcings::prepare_meteo_forcings_data().

17.18.2.42 l1_pre_weights

```
real(dp), dimension(:, :, :), allocatable, public mo_global_variables::l1_pre_weights
```

Definition at line 100 of file mo_global_variables.f90.

Referenced by mo_mhm_eval::mhm_eval(), and mo_meteo_forcings::prepare_meteo_forcings_data().

17.18.2.43 l1_preeffect

```
real(dp), dimension(:), allocatable, public mo_global_variables::l1_preeffect
```

Definition at line 160 of file mo_global_variables.f90.

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().

17.18.2.44 l1_rain

```
real(dp), dimension(:), allocatable, public mo_global_variables::l1_rain
```

Definition at line 161 of file mo_global_variables.f90.

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().

17.18.2.45 l1_runoffseal

```
real(dp), dimension(:), allocatable, public mo_global_variables::l1_runoffseal
```

Definition at line 162 of file mo_global_variables.f90.

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().

17.18.2.46 l1_satstw

```
real(dp), dimension(:), allocatable, public mo_global_variables::l1_satstw
```

Definition at line 145 of file mo_global_variables.f90.

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().

17.18.2.47 l1_sealstw

```
real(dp), dimension(:), allocatable, public mo_global_variables::l1_sealstw
```

Definition at line 142 of file mo_global_variables.f90.

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().

17.18.2.48 l1_slowrunoff

```
real(dp), dimension(:), allocatable, public mo_global_variables::l1_slowrunoff
```

Definition at line 163 of file mo_global_variables.f90.

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().

17.18.2.49 l1_sm

```
real(dp), dimension(:, :), allocatable, public mo_global_variables::l1_sm
```

Definition at line 116 of file mo_global_variables.f90.

17.18.2.50 l1_sm_mask

```
logical, dimension(:, :), allocatable, public mo_global_variables::l1_sm_mask
```

Definition at line 117 of file mo_global_variables.f90.

17.18.2.51 l1_smobs

```
type(optidata), dimension(:), allocatable, public mo_global_variables::l1_smobs
```

Definition at line 125 of file mo_global_variables.f90.

Referenced by mo_mhm_eval::mhm_eval(), mo_mhm_read_config::mhm_read_config(), 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(), and mo_objective_function::objective_sm_sse↩ _standard_score().

17.18.2.52 l1_snow

```
real(dp), dimension(:), allocatable, public mo_global_variables::l1_snow
```

Definition at line 164 of file mo_global_variables.f90.

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().

17.18.2.53 l1_snowpack

```
real(dp), dimension(:), allocatable, public mo_global_variables::l1_snowpack
```

Definition at line 141 of file mo_global_variables.f90.

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().

17.18.2.54 l1_soilmoist

```
real(dp), dimension(:, :), allocatable, public mo_global_variables::l1_soilmoist
```

Definition at line 143 of file mo_global_variables.f90.

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().

17.18.2.55 l1_ssrd

```
real(dp), dimension(:, :), allocatable, public mo_global_variables::l1_ssrd
```

Definition at line 110 of file mo_global_variables.f90.

Referenced by mo_mhm_eval::mhm_eval(), and mo_meteo_forcings::prepare_meteo_forcings_data().

17.18.2.56 l1_strd

```
real(dp), dimension(:, :), allocatable, public mo_global_variables::l1_strd
```

Definition at line 111 of file mo_global_variables.f90.

Referenced by mo_mhm_eval::mhm_eval(), and mo_meteo_forcings::prepare_meteo_forcings_data().

17.18.2.57 l1_tann

```
real(dp), dimension(:, :), allocatable, public mo_global_variables::l1_tann
```

Definition at line 112 of file mo_global_variables.f90.

Referenced by mo_mhm_eval::mhm_eval(), and mo_meteo_forcings::prepare_meteo_forcings_data().

17.18.2.58 l1_temp

```
real(dp), dimension(:, :), allocatable, public mo_global_variables::l1_temp
```

Definition at line 102 of file mo_global_variables.f90.

Referenced by mo_mhm_eval::mhm_eval(), and mo_meteo_forcings::prepare_meteo_forcings_data().

17.18.2.59 l1_temp_weights

```
real(dp), dimension(:, :, :), allocatable, public mo_global_variables::l1_temp_weights
```

Definition at line 98 of file mo_global_variables.f90.

Referenced by mo_mhm_eval::mhm_eval(), and mo_meteo_forcings::prepare_meteo_forcings_data().

17.18.2.60 l1_throughfall

```
real(dp), dimension(:), allocatable, public mo_global_variables::l1_throughfall
```

Definition at line 165 of file mo_global_variables.f90.

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().

17.18.2.61 l1_tmax

```
real(dp), dimension(:, :), allocatable, public mo_global_variables::l1_tmax
```

Definition at line 105 of file mo_global_variables.f90.

Referenced by mo_mhm_eval::mhm_eval(), and mo_meteo_forcings::prepare_meteo_forcings_data().

17.18.2.62 l1_tmin

```
real(dp), dimension(:, :), allocatable, public mo_global_variables::l1_tmin
```

Definition at line 104 of file mo_global_variables.f90.

Referenced by mo_mhm_eval::mhm_eval(), and mo_meteo_forcings::prepare_meteo_forcings_data().

17.18.2.63 l1_total_runoff

```
real(dp), dimension(:), allocatable, public mo_global_variables::l1_total_runoff
```

Definition at line 166 of file mo_global_variables.f90.

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().

17.18.2.64 l1_twsaobs

```
type(optidata), dimension(:), allocatable, public mo_global_variables::l1_twsaobs
```

Definition at line 131 of file mo_global_variables.f90.

Referenced by mo_objective_function::create_domain_avg_tws(), mo_mhm_eval::mhm_eval(), mo_mhm_read↩_config::mhm_read_config(), mo_objective_function::objective_kge_q_rmse_tws(), and mo_objective_function↩::objective_q_et_tws_kge_catchment_avg().

17.18.2.65 l1_unsatstw

```
real(dp), dimension(:), allocatable, public mo_global_variables::l1_unsatstw
```

Definition at line 144 of file mo_global_variables.f90.

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().

17.18.2.66 l1_windspeed

```
real(dp), dimension(:, :), allocatable, public mo_global_variables::l1_windspeed
```

Definition at line 108 of file mo_global_variables.f90.

Referenced by mo_mhm_eval::mhm_eval(), and mo_meteo_forcings::prepare_meteo_forcings_data().

17.18.2.67 level2

```
type(grid), dimension(:), allocatable, public mo_global_variables::level2
```

Definition at line 90 of file mo_global_variables.f90.

Referenced by mo_meteo_forcings::meteo_forcings_wrapper(), mo_meteo_forcings::meteo_weights_wrapper(), and mo_startup::mhm_initialize().

17.18.2.68 neutron_integral_afast

```
real(dp), dimension(:), allocatable, public mo_global_variables::neutron_integral_afast
```

Definition at line 190 of file mo_global_variables.f90.

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

17.18.2.69 nsoilhorizons_sm_input

```
integer(i4) mo_global_variables::nsoilhorizons_sm_input
```

Definition at line 123 of file mo_global_variables.f90.

Referenced by mo_mhm_eval::mhm_eval(), and mo_mhm_read_config::mhm_read_config().

17.18.2.70 output_deflate_level

```
integer(i4) mo_global_variables::output_deflate_level
```

Definition at line 55 of file mo_global_variables.f90.

Referenced by mo_mhm_read_config::mhm_read_config(), and mo_write_fluxes_states::newoutputvariable().

17.18.2.71 output_double_precision

```
logical mo_global_variables::output_double_precision
```

Definition at line 56 of file mo_global_variables.f90.

Referenced by mo_write_fluxes_states::createoutputfile(), and mo_mhm_read_config::mhm_read_config().

17.18.2.72 outputflxstate

```
logical, dimension(noutflxstate) mo_global_variables::outputflxstate
```

Definition at line 58 of file mo_global_variables.f90.

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().

17.18.2.73 read_meteo_weights

```
logical, public mo_global_variables::read_meteo_weights
```

Definition at line 64 of file mo_global_variables.f90.

Referenced by mo_mhm_eval::mhm_eval(), mo_mhm_read_config::mhm_read_config(), and mo_meteo_forcings↩ ::prepare_meteo_forcings_data().

17.18.2.74 routingstates

```
integer(i4), parameter, public mo_global_variables::routingstates = 2
```

Definition at line 85 of file mo_global_variables.f90.

17.18.2.75 timestep_model_inputs

```
integer(i4), dimension(:), allocatable, public mo_global_variables::timestep_model_inputs
```

Definition at line 63 of file mo_global_variables.f90.

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().

17.18.2.76 timestep_model_outputs

```
integer(i4) mo_global_variables::timestep_model_outputs
```

Definition at line 57 of file mo_global_variables.f90.

Referenced by mo_write_fluxes_states::fluxesunit(), mo_mhm_eval::mhm_eval(), and mo_mhm_read_config↩ ::mhm_read_config().

17.19 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_domain_indices (grids, indices)

  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.

17.19.1 Detailed Description

TODO: add description.

TODO: add description

Authors

Robert Schweppe

Date

Jun 2018

17.19.2 Function/Subroutine Documentation

17.19.2.1 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

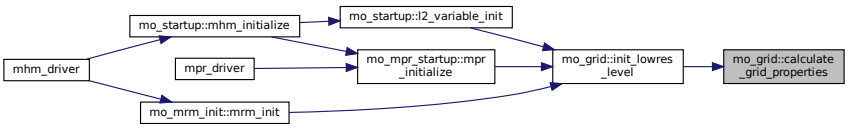
Date

Feb 2013

Definition at line 489 of file mo_grid.f90.

Referenced by init_lowres_level().

Here is the caller graph for this function:



17.19.2.2 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 domain and level

Parameters

in	*type(Grid) :: level*	-> grid reference
out	*real(dp), dimension(:, :) :: lat, lon*	
out	*real(dp), dimension(:, :) :: lat, lon*	

Authors

Matthias Zink

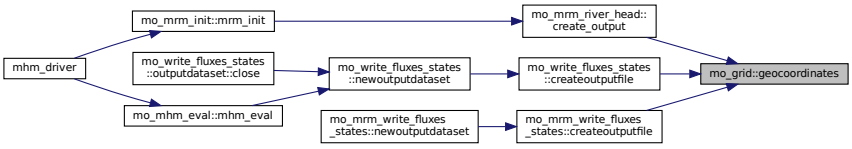
Date

Apr 2013

Definition at line 433 of file mo_grid.f90.

Referenced by mo_mrm_river_head::create_output(), mo_write_fluxes_states::createoutputfile(), and mo_mrm_↩ write_fluxes_states::createoutputfile().

Here is the caller graph for this function:



17.19.2.3 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

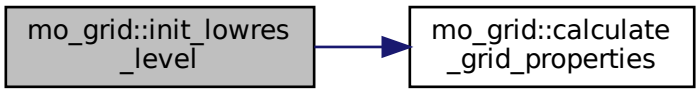
Jan 2013

Definition at line 59 of file mo_grid.f90.

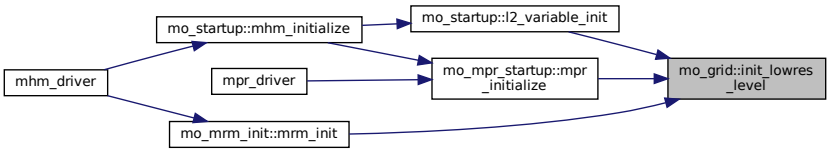
References calculate_grid_properties(), mo_common_constants::nodata_dp, and mo_common_constants::nodata_i4.

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:



17.19.2.4 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

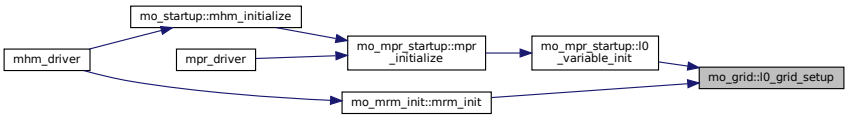
Jan 2013

Definition at line 268 of file mo_grid.f90.

References mo_common_variables::iflag_cordinate_sys.

Referenced by mo_mpr_startup::l0_variable_init(), and mo_mrm_init::mrm_init().

Here is the caller graph for this function:



17.19.2.5 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 domain and level

Parameters

in	*type(Grid) :: level*	-> grid reference
out	*real(dp), dimension(:) :: x, y*	
out	*real(dp), dimension(:) :: x, y*	

Authors

Matthias Zink

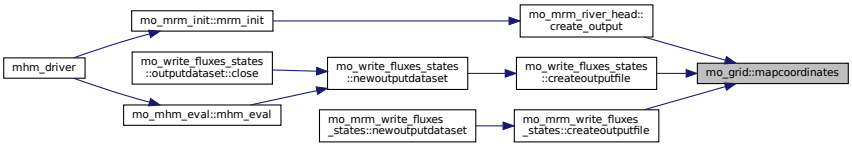
Date

Apr 2013

Definition at line 370 of file mo_grid.f90.

Referenced by mo_mrm_river_head::create_output(), mo_write_fluxes_states::createoutputfile(), and mo_mrm_↩ write_fluxes_states::createoutputfile().

Here is the caller graph for this function:



17.19.2.6 set_domain_indices()

```
subroutine, public mo_grid::set_domain_indices (
             type(grid), dimension(:), intent(inout) grids,
             integer(i4), dimension(:), intent(in), optional indices )
```

TODO: add description.

TODO: add description

Parameters

in,out	*type(Grid), dimension(:) :: grids*	

Authors

Robert Schweppe

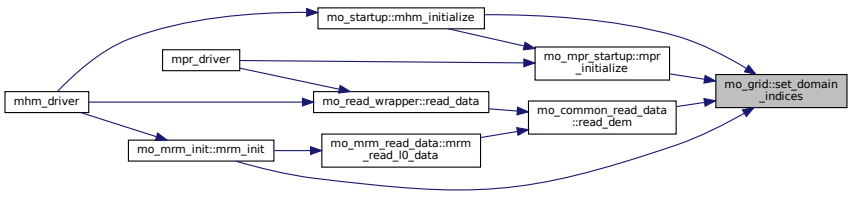
Date

Jun 2018

Definition at line 205 of file mo_grid.f90.

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:



17.20 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.

17.20.1 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

17.20.2 Function/Subroutine Documentation

17.20.2.1 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 domain. 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

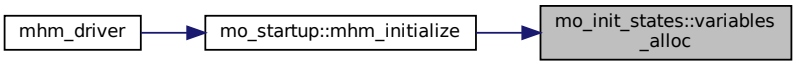
Jan 2013

Definition at line 62 of file mo_init_states.f90.

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:



17.20.2.2 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

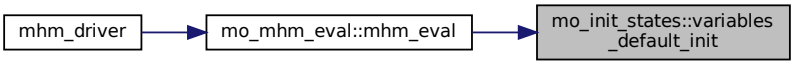
Definition at line 192 of file mo_init_states.f90.

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:



17.21 mo_meteo_forcings Module Reference

Prepare meteorological forcings data for mHM.

Functions/Subroutines

- subroutine, public prepare_meteo_forcings_data (iDomain, domainID, tt)

  Prepare meteorological forcings data for a given variable.
- subroutine meteo_forcings_wrapper (iDomain, dataPath, inputFormat, dataOut1, lower, upper, ncvarName)

  Prepare meteorological forcings data for mHM at Level-1.
- subroutine meteo_weights_wrapper (iDomain, read_meteo_weights, dataPath, dataOut1, lower, upper, ncvarName)

  Prepare weights for meteorological forcings data for mHM at Level-1.
- subroutine chunk_config (iDomain, tt, read_flag, readPer)

  determines the start date, end date, and read_flag given Domain id and current timestep
- logical function is_read (iDomain, tt)

  evaluate whether new chunk should be read at this timestep
- subroutine chunk_size (iDomain, tt, readPer)

  calculate beginning and end of read Period, i.e. that is length of current chunk to read

17.21.1 Detailed Description

Prepare meteorological forcings data for mHM.

Prepare meteorological forcings data for mHM.

Authors

Rohini Kumar

Date

Jan 2012

17.21.2 Function/Subroutine Documentation

17.21.2.1 chunk_config()

```
subroutine mo_meteo_forcings::chunk_config (
             integer(i4), intent(in) iDomain,
             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 Domain id and current timestep

TODO: add description

Parameters

in	*integer(i4) :: iDomain*	current Domain
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

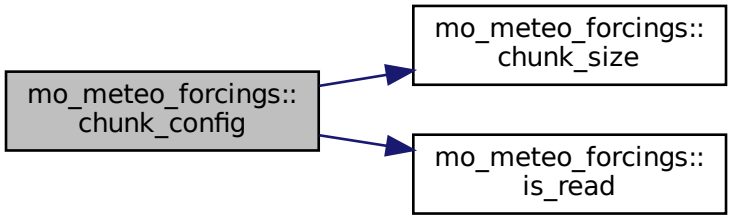
Jun 2014

Definition at line 584 of file mo_meteo_forcings.f90.

References chunk_size(), 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:

17.21.2.2 chunk_size()

```
subroutine mo_meteo_forcings::chunk_size (
             integer(i4), intent(in) iDomain,
             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) :: iDomain*	current Domain to process
in	*integer(i4) :: tt*	current time step
out	*type(period) :: readPer*	start and end dates of read Period

Authors

Stephan Thober

Date

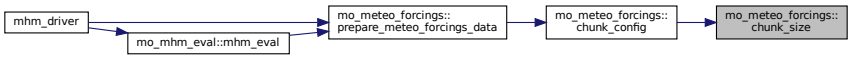
Jun 2014

Definition at line 759 of file mo_meteo_forcings.f90.

References 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 caller graph for this function:



17.21.2.3 is_read()

```
logical function mo_meteo_forcings::is_read (
            integer(i4), intent(in) iDomain,
            integer(i4), intent(in) tt )
```

evaluate whether new chunk should be read at this timestep

TODO: add description

Parameters

in	*integer(i4) :: iDomain*	current Domain
in	*integer(i4) :: tt*	current time step

Authors

Stephan Thober

Date

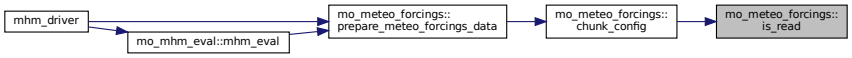
Jun 2014

Definition at line 648 of file mo_meteo_forcings.f90.

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_inputs.

Referenced by chunk_config().

Here is the caller graph for this function:



17.21.2.4 meteo_forcings_wrapper()

```
subroutine mo_meteo_forcings::meteo_forcings_wrapper (
            integer(i4), intent(in) iDomain,
            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 Domain 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 Domain to their respective global ones

Parameters

in	*integer(i4) :: iDomain*	Domain Id
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

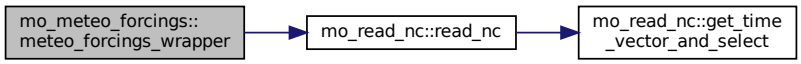
Jan 2013

Definition at line 267 of file mo_meteo_forcings.f90.

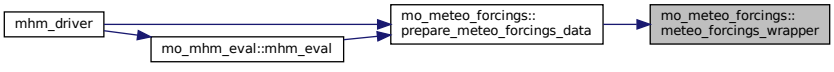
References mo_common_variables::level1, mo_global_variables::level2, mo_common_constants::nodata_dp, mo_read_nc::read_nc(), and mo_common_mhm_mrm_variables::readper.

Referenced by prepare_meteo_forcings_data().

Here is the call graph for this function:



Here is the caller graph for this function:



17.21.2.5 meteo_weights_wrapper()

```
subroutine mo_meteo_forcings::meteo_weights_wrapper (
            integer(i4), intent(in) iDomain,
            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 Domain 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 Domain to their respective global ones

Parameters

in	*integer(i4) :: iDomain*	Domain 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

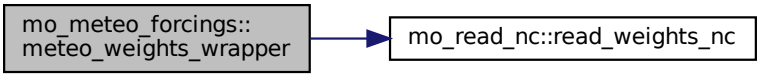
Jan 2017

Definition at line 432 of file mo_meteo_forcings.f90.

References mo_common_variables::level1, mo_global_variables::level2, mo_common_constants::nodata_dp, and mo_read_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:

17.21.2.6 prepare_meteo_forcings_data()

```
subroutine, public mo_meteo_forcings::prepare_meteo_forcings_data (
             integer(i4), intent(in) iDomain,
             integer(i4), intent(in) domainID,
             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) :: iDomain*	Domain Id
in	*integer(i4) :: tt*	current timestep

Authors

Rohini Kumar

Date

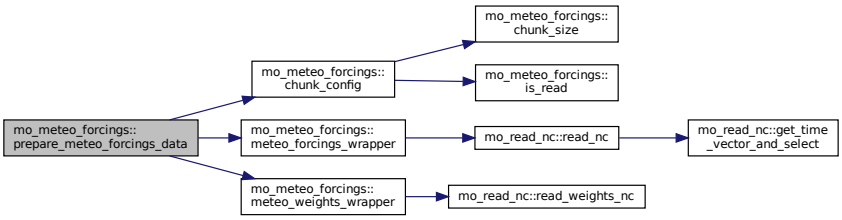
Jan 2013

Definition at line 63 of file mo_meteo_forcings.f90.

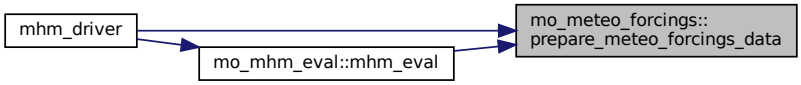
References chunk_config(), mo_global_variables::dirabsvappressure, mo_global_variables::dirmaxtemperature, mo_global_variables::dirmintemperature, mo_global_variables::dirnetradiation, mo_global_variables::dirprecipitation, mo_global_variables::dirradiation, mo_global_variables::dirreferenceet, mo_global_variables::dirtemperature, mo↩ _global_variables::dirwindspeed, mo_common_variables::domainmeta, 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_ssrd, mo_global_variables::l1_strd, mo_global_variables::l1_tann, 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, meteo_forcings_wrapper(), meteo_weights_wrapper(), mo_common_variables↩ ::processmatrix, mo_global_variables::read_meteo_weights, mo_common_mhm_mrm_variables::readper, 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:



17.22 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.

17.22.1 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

Process	Name	Flag	Description
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.
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

17.22.2 Function/Subroutine Documentation

17.22.2.1 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 domain 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 TS-1] estimated PET (if PET is input = corrected values (fAsp$*$PET))

Parameters

in,out	*real(dp), dimension(:, :) :: aet_soil*	actual ET
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

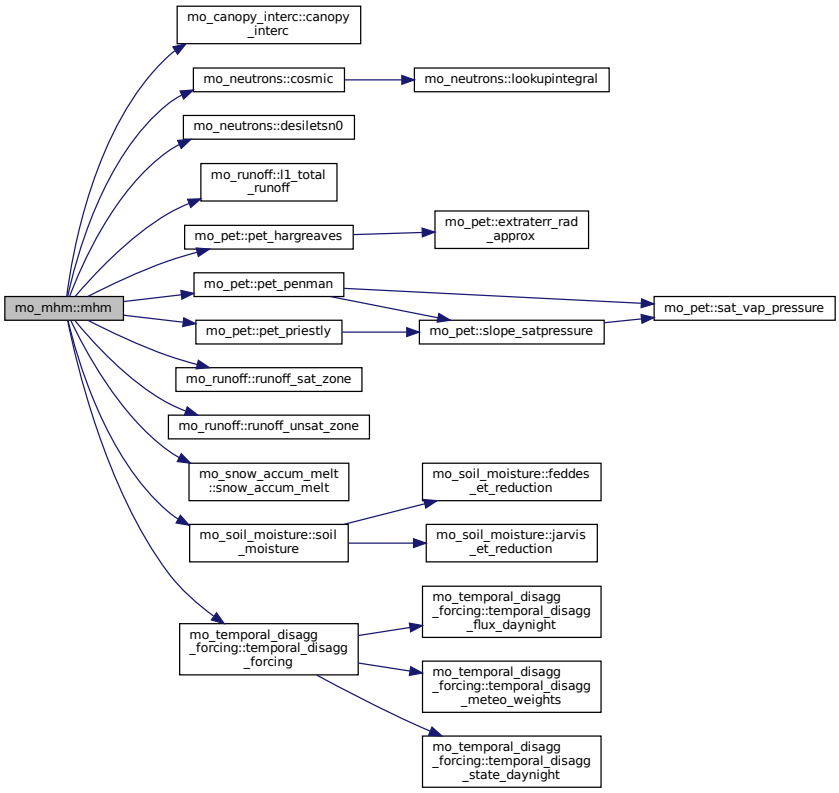
Dec 2012

Definition at line 200 of file mo_mhm.f90.

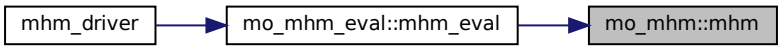
References mo_canopy_interc::canopy_interc(), mo_neutrons::cosmic(), mo_neutrons::desiletsn0(), mo_mhm_constants::harsamconst, mo_runoff::l1_total_runoff(), mo_pet::pet_hargreaves(), mo_pet::pet_penman(), mo_pet::pet_priestly(), mo_runoff::runoff_sat_zone(), mo_runoff::runoff_unsat_zone(), mo_snow_accum_melt::snow_accum_melt(), 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:



17.23 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 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

17.23.1 Detailed Description

Provides mHM specific constants.

Provides mHM specific constants such as flood plain elevation.

Authors

Matthias Cuntz

Date

Nov 2011

17.23.2 Variable Documentation

17.23.2.1 c1_initstatesm

```
real(dp), parameter, public mo_mhm_constants::c1_initstatesm = 0.25_dp
```

Definition at line 30 of file mo_mhm_constants.f90.

17.23.2.2 cosmic_alpha

```
real(dp), parameter, public mo_mhm_constants::cosmic_alpha = 0.2392421548_dp
```

Definition at line 59 of file mo_mhm_constants.f90.

Referenced by mo_neutrons::cosmic().

17.23.2.3 cosmic_bd

```
real(dp), parameter, public mo_mhm_constants::cosmic_bd = 1.4020_dp
```

Neutrons and moisture: COSMIC, Shuttleworth et al. 2013.

Definition at line 56 of file mo_mhm_constants.f90.

Referenced by mo_neutrons::cosmic().

17.23.2.4 cosmic_l1

```
real(dp), parameter, public mo_mhm_constants::cosmic_l1 = 161.98621864_dp
```

Definition at line 60 of file mo_mhm_constants.f90.

Referenced by mo_neutrons::cosmic().

17.23.2.5 cosmic_l2

```
real(dp), parameter, public mo_mhm_constants::cosmic_l2 = 129.14558985_dp
```

Definition at line 61 of file mo_mhm_constants.f90.

Referenced by mo_neutrons::cosmic().

17.23.2.6 cosmic_l3

```
real(dp), parameter, public mo_mhm_constants::cosmic_l3 = 107.82204562_dp
```

Definition at line 62 of file mo_mhm_constants.f90.

Referenced by mo_neutrons::cosmic().

17.23.2.7 cosmic_l4

```
real(dp), parameter, public mo_mhm_constants::cosmic_l4 = 3.1627190566_dp
```

Definition at line 63 of file mo_mhm_constants.f90.

Referenced by mo_neutrons::cosmic().

17.23.2.8 cosmic_n

```
real(dp), parameter, public mo_mhm_constants::cosmic_n = 348.33_dp
```

Definition at line 58 of file mo_mhm_constants.f90.

Referenced by mo_neutrons::cosmic().

17.23.2.9 cosmic_vwclat

```
real(dp), parameter, public mo_mhm_constants::cosmic_vwclat = 0.0753_dp
```

Definition at line 57 of file mo_mhm_constants.f90.

Referenced by mo_neutrons::cosmic().

17.23.2.10 desilets_a0

```
real(dp), parameter, public mo_mhm_constants::desilets_a0 = 0.0808_dp
```

Neutrons and moisture: N0 formula, Desilets et al. 2010.

Definition at line 51 of file mo_mhm_constants.f90.

Referenced by mo_neutrons::desiletsn0().

17.23.2.11 desilets_a1

```
real(dp), parameter, public mo_mhm_constants::desilets_a1 = 0.372_dp
```

Definition at line 52 of file mo_mhm_constants.f90.

Referenced by mo_neutrons::desiletsn0().

17.23.2.12 desilets_a2

```
real(dp), parameter, public mo_mhm_constants::desilets_a2 = 0.115_dp
```

Definition at line 53 of file mo_mhm_constants.f90.

Referenced by mo_neutrons::desiletsn0().

17.23.2.13 duffiedelta1

```
real(dp), parameter, public mo_mhm_constants::duffiedelta1 = 0.4090_dp
```

Definition at line 40 of file mo_mhm_constants.f90.

Referenced by mo_pet::extraterr_rad_approx().

17.23.2.14 duffiedelta2

```
real(dp), parameter, public mo_mhm_constants::duffiedelta2 = 1.3900_dp
```

Definition at line 41 of file mo_mhm_constants.f90.

Referenced by mo_pet::extraterr_rad_approx().

17.23.2.15 duffiedr

```
real(dp), parameter, public mo_mhm_constants::duffiedr = 0.0330_dp
```

Definition at line 39 of file mo_mhm_constants.f90.

Referenced by mo_pet::extraterr_rad_approx().

17.23.2.16 h2odens

```
real(dp), parameter, public mo_mhm_constants::h2odens = 1000.0_dp
```

Definition at line 23 of file mo_mhm_constants.f90.

Referenced by mo_neutrons::cosmic(), mo_mrm_riv_temp_class::finalize_source_e(), and mo_mrm_riv_temp_↩class::get_e_io().

17.23.2.17 harsamconst

```
real(dp), parameter, public mo_mhm_constants::harsamconst = 17.800_dp
```

Hargreaves-Samani ref. ET formula [deg C].

Definition at line 36 of file mo_mhm_constants.f90.

Referenced by mo_mhm::mhm().

17.23.2.18 noutflxstate

```
integer(i4), parameter, public mo_mhm_constants::noutflxstate = 20_i4
```

Definition at line 33 of file mo_mhm_constants.f90.

17.23.2.19 p2_initstatefluxes

```
real(dp), parameter, public mo_mhm_constants::p2_initstatefluxes = 15.00_dp
```

Definition at line 26 of file mo_mhm_constants.f90.

17.23.2.20 p3_initstatefluxes

```
real(dp), parameter, public mo_mhm_constants::p3_initstatefluxes = 10.00_dp
```

Definition at line 27 of file mo_mhm_constants.f90.

17.23.2.21 p4_initstatefluxes

```
real(dp), parameter, public mo_mhm_constants::p4_initstatefluxes = 75.00_dp
```

Definition at line 28 of file mo_mhm_constants.f90.

17.23.2.22 p5_initstatefluxes

```
real(dp), parameter, public mo_mhm_constants::p5_initstatefluxes = 1500.00_dp
```

Definition at line 29 of file mo_mhm_constants.f90.

17.23.2.23 satpressureslope1

```
real(dp), parameter, public mo_mhm_constants::satpressureslope1 = 4098.0_dp
```

calculation of the slope of the saturation vapour pressure curve following Tetens

Definition at line 48 of file mo_mhm_constants.f90.

Referenced by mo_pet::slope_satpressure().

17.23.2.24 tetens_c1

```
real(dp), parameter, public mo_mhm_constants::tetens_c1 = 0.6108_dp
```

Tetens's formula to calculate saturated vapour pressure.

Definition at line 44 of file mo_mhm_constants.f90.

Referenced by mo_pet::sat_vap_pressure().

17.23.2.25 tetens_c2

```
real(dp), parameter, public mo_mhm_constants::tetens_c2 = 17.270_dp
```

Definition at line 45 of file mo_mhm_constants.f90.

Referenced by mo_pet::sat_vap_pressure().

17.23.2.26 tetens_c3

```
real(dp), parameter, public mo_mhm_constants::tetens_c3 = 237.30_dp
```

Definition at line 46 of file mo_mhm_constants.f90.

Referenced by mo_pet::sat_vap_pressure(), and mo_pet::slope_satpressure().

17.24 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, opti_domain_indices, runoff, smOptiSim, neutronsOptiSim, et↩ OptiSim, twsOptiSim)

  Runs mhm with a specific parameter set and returns required variables, e.g. runoff.

17.24.1 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

17.24.2 Function/Subroutine Documentation

17.24.2.1 mhm_eval()

```
subroutine, public mo_mhm_eval::mhm_eval (
             real(dp), dimension(:), intent(in) parameterset,
             integer(i4), dimension(:), intent(in), optional opti_domain_indices,
             real(dp), dimension(:, :), intent(out), optional, allocatable runoff,
             type(optidata_sim), dimension(:), intent(inout), optional smOptiSim,
             type(optidata_sim), dimension(:), intent(inout), optional neutronsOptiSim,
             type(optidata_sim), dimension(:), intent(inout), optional etOptiSim,
             type(optidata_sim), dimension(:), intent(inout), optional twsOptiSim )
```

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 Domains concatenated),DIMENSION [nCells, nTimeSteps] time series, DIMENSION [nTimeSteps, nDomains]
out	*real(dp), dimension(:, :), optional :: neutrons_opti*	dim1=ncells, dim2=time
out	*real(dp), dimension(:, :), optional :: et_opti*	returns evapotranspiration time series for
out	*real(dp), dimension(:, :), optional :: tws_opti*	returns tws time series all grid cells (of multiple Domains concatenated),DIMENSION [nCells, nTimeSteps]

Authors

Juliane Mai, Rohini Kumar

Date

Feb 2013

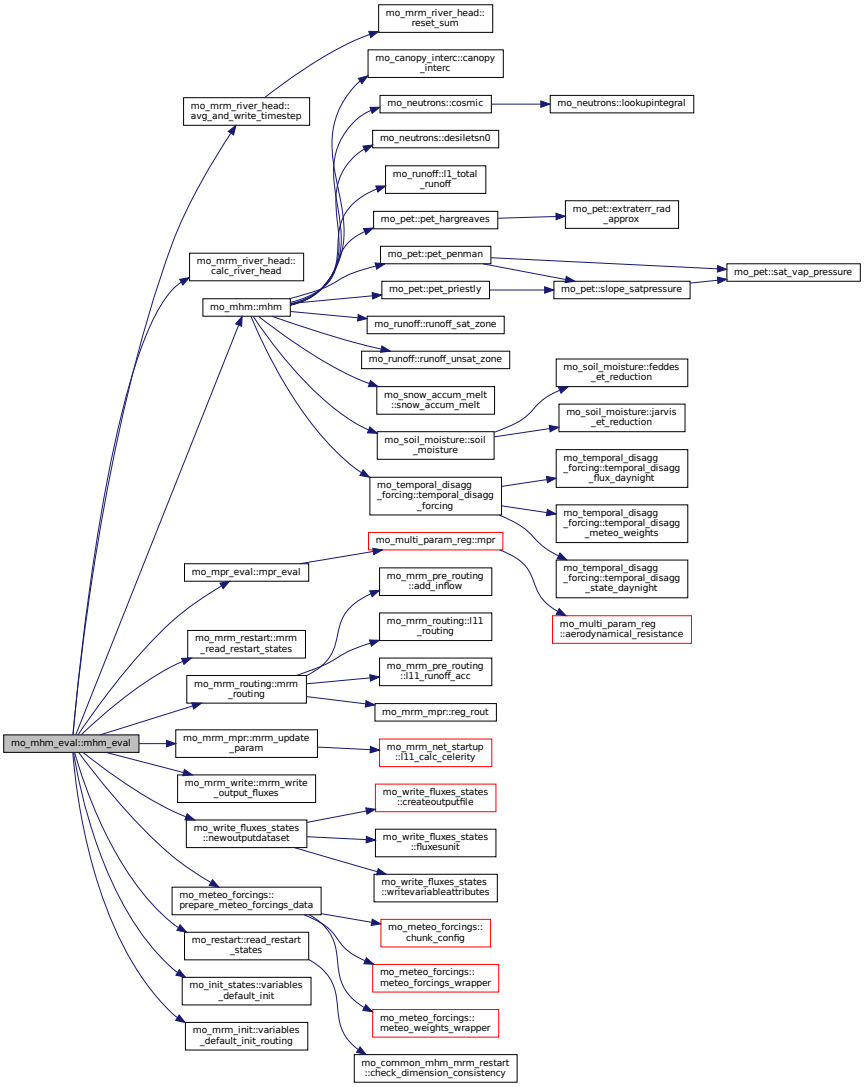
Definition at line 83 of file mo_mhm_eval.f90.

References mo_mrm_river_head::avg_and_write_timestep(), mo_common_mhm_mrm_variables::c2tstu, mo_mrm_river_head::calc_river_head(), mo_mrm_global_variables::domain_mrm, mo_common_variables::domainmeta, mo_global_variables::evap_coeff, mo_global_variables::fday_pet, mo_global_variables::fday_prec, mo_global_variables::fday_ssrd, mo_global_variables::fday_strd, mo_global_variables::fday_temp, mo_global_variables::fnight_pet, mo_global_variables::fnight_prec, mo_global_variables::fnight_ssrd, mo_global_variables::fnight_strd, mo_global_variables::fnight_temp, mo_mrm_global_variables::gw_coupling, mo_mpr_global_variables::horizondepth_mhm, mo_mrm_global_variables::inflowgauge, 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_global_variables::l1_etobs, 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_neutronsobs, 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_smobs, 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_global_variables::l1_ssrd, mo_global_variables::

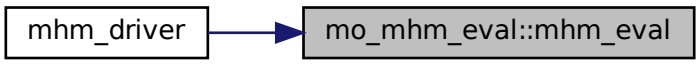
::l1_strd, mo_mpr_global_variables::l1_surfresist, mo_global_variables::l1_tann, mo_global_variables::l1_↩
temp, mo_global_variables::l1_temp_weights, mo_mpr_global_variables::l1_tempthresh, mo_global_variables↩
::l1_throughfall, mo_global_variables::l1_tmax, mo_global_variables::l1_tmin, mo_global_variables::l1_total↩
_runoff, mo_global_variables::l1_twsaobs, 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_mhm↩
::mhm(), mo_common_mhm_mrm_variables::mhmfilerestartin, 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_mhm_mrm_variables↩
::mrmfilerestartin, mo_global_variables::neutron_integral_afast, mo_write_fluxes_states::newoutputdataset(), mo↩
_mpr_global_variables::nsoilhorizons_mhm, mo_global_variables::nsoilhorizons_sm_input, 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_mrm_↩
global_variables::riv_temp_pcs, mo_common_mhm_mrm_variables::simper, mo_common_mhm_mrm_variables↩
::timestep, mo_global_variables::timestep_model_inputs, mo_global_variables::timestep_model_outputs, mo↩
_mrm_global_variables::timestep_model_outputs_mrm, 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:



17.25 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.

17.25.1 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

17.25.2 Function/Subroutine Documentation

17.25.2.1 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

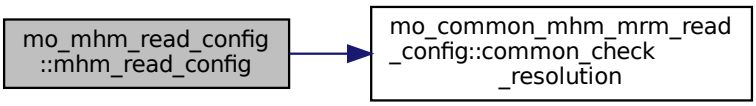
Dec 2012

Definition at line 89 of file mo_mhm_read_config.f90.

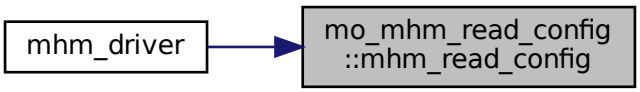
References mo_common_mhm_mrm_read_config::common_check_resolution(), mo_global_variables::dirabsvappressure, mo_global_variables::dirmaxtemperature, mo_global_variables::dirmintemperature, mo_global_variables::dirnetradiation, mo_global_variables::dirprecipitation, mo_global_variables::dirradiation, mo_global_variables::dirreferenceet, mo_global_variables::dirtemperature, mo_global_variables::dirwindspeed, mo_common_variables::domainmeta, mo_global_variables::evap_coeff, mo_global_variables::fday_pet, mo_global_variables::fday_prec, mo_global_variables::fday_ssrd, mo_global_variables::fday_strd, mo_global_variables::fday_temp, mo_file::file_defoutput, mo_global_variables::fnight_pet, mo_global_variables::fnight_prec, mo_global_variables::fnight_ssrd, mo_global_variables::fnight_strd, mo_global_variables::fnight_temp, mo_global_variables::inputformat_meteo_forcings, mo_global_variables::l1_etobs, mo_global_variables::l1_neutronsobs, mo_global_variables::l1_smobs, mo_global_variables::l1_twsaobs, mo_common_constants::maxnodomains, mo_mpr_constants::maxnosoilhorizons, mo_common_constants::nodata_i4, mo_mpr_global_variables::nsoilhorizons_mhm, mo_global_variables::nsoilhorizons_sm_input, mo_common_mhm_mrm_variables::opti_function, mo_common_mhm_mrm_variables::optimize, mo_global_variables::output_deflate_level, mo_global_variables::output_double_precision, mo_global_variables::outputflxstate, mo_common_variables::processmatrix, mo_global_variables::read_meteo_weights, mo_global_variables::timestep_model_inputs, mo_global_variables::timestep_model_outputs, and mo_file::udefoutput.

Referenced by mhm_driver().

Here is the call graph for this function:



Here is the caller graph for this function:



17.26 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

17.26.1 Detailed Description

Provides MPR specific constants.

Provides MPR specific constants such as flood plain elevation.

Authors

Matthias Cuntz

Date

Nov 2011

17.26.2 Variable Documentation

17.26.2.1 bulkdens_orgmatter

```
real(dp), parameter, public mo_mpr_constants::bulkdens_orgmatter = 0.224_dp
```

Definition at line 35 of file mo_mpr_constants.f90.

Referenced by mo_mpr_soilmoist::mpr_sm().

17.26.2.2 c1_initstatesm

```
real(dp), parameter, public mo_mpr_constants::c1_initstatesm = 0.25_dp
```

Definition at line 31 of file mo_mpr_constants.f90.

Referenced by mo_init_states::variables_alloc(), and mo_init_states::variables_default_init().

17.26.2.3 field_cap_c1

```
real(dp), parameter, public mo_mpr_constants::field_cap_c1 = -0.60_dp
```

Definition at line 37 of file mo_mpr_constants.f90.

Referenced by mo_mpr_soilmoist::field_cap().

17.26.2.4 field_cap_c2

```
real(dp), parameter, public mo_mpr_constants::field_cap_c2 = 2.0_dp
```

Definition at line 38 of file mo_mpr_constants.f90.

Referenced by mo_mpr_soilmoist::field_cap().

17.26.2.5 karman

```
real(dp), parameter, public mo_mpr_constants::karman = 0.41_dp
```

von karman constant

Definition at line 68 of file mo_mpr_constants.f90.

Referenced by mo_multi_param_reg::aerodynamical_resistance().

17.26.2.6 ks_c

```
real(dp), parameter, public mo_mpr_constants::ks_c = 10.0_dp
```

Definition at line 60 of file mo_mpr_constants.f90.

Referenced by mo_mpr_soilmoist::hydro_cond().

17.26.2.7 lai_factor_surfresi

```
real(dp), parameter, public mo_mpr_constants::lai_factor_surfresi = 0.3_dp
```

LAI factor for bulk surface resistance formulation.

Definition at line 71 of file mo_mpr_constants.f90.

Referenced by mo_mpr_pet::bulksurface_resistance().

17.26.2.8 lai_offset_surfresi

```
real(dp), parameter, public mo_mpr_constants::lai_offset_surfresi = 1.2_dp
```

LAI offset for bulk surface resistance formulation.

Definition at line 73 of file mo_mpr_constants.f90.

Referenced by mo_mpr_pet::bulksurface_resistance().

17.26.2.9 max_surfresist

```
real(dp), parameter, public mo_mpr_constants::max_surfresist = 250.0_dp
```

maximum bulk surface resistance

Definition at line 75 of file mo_mpr_constants.f90.

Referenced by mo_mpr_pet::bulksurface_resistance().

17.26.2.10 maxgeounit

```
integer(i4), parameter, public mo_mpr_constants::maxgeounit = 25_i4
```

Definition at line 23 of file mo_mpr_constants.f90.

Referenced by mo_mpr_read_config::mpr_read_config().

17.26.2.11 maxnosoilhorizons

```
integer(i4), parameter, public mo_mpr_constants::maxnosoilhorizons = 10_i4
```

Definition at line 24 of file mo_mpr_constants.f90.

Referenced by mo_mhm_read_config::mhm_read_config(), and mo_mpr_read_config::mpr_read_config().

17.26.2.12 nlcover_class

```
integer(i4), parameter, public mo_mpr_constants::nlcover_class = 3_i4
```

Definition at line 22 of file mo_mpr_constants.f90.

Referenced by mo_soil_database::read_soil_lut().

17.26.2.13 p2_initstatefluxes

```
real(dp), parameter, public mo_mpr_constants::p2_initstatefluxes = 15.00_dp
```

Definition at line 27 of file mo_mpr_constants.f90.

Referenced by mo_init_states::variables_alloc(), and mo_init_states::variables_default_init().

17.26.2.14 p3_initstatefluxes

```
real(dp), parameter, public mo_mpr_constants::p3_initstatefluxes = 10.00_dp
```

Definition at line 28 of file mo_mpr_constants.f90.

Referenced by mo_init_states::variables_alloc(), and mo_init_states::variables_default_init().

17.26.2.15 p4_initstatefluxes

```
real(dp), parameter, public mo_mpr_constants::p4_initstatefluxes = 75.00_dp
```

Definition at line 29 of file mo_mpr_constants.f90.

Referenced by mo_init_states::variables_alloc(), and mo_init_states::variables_default_init().

17.26.2.16 p5_initstatefluxes

```
real(dp), parameter, public mo_mpr_constants::p5_initstatefluxes = 1500.00_dp
```

Definition at line 30 of file mo_mpr_constants.f90.

Referenced by mo_init_states::variables_alloc(), and mo_init_states::variables_default_init().

17.26.2.17 pwp_c

```
real(dp), parameter, public mo_mpr_constants::pwp_c = 1.0_dp
```

Definition at line 62 of file mo_mpr_constants.f90.

Referenced by mo_mpr_soilmoist::pwp().

17.26.2.18 pwp_matpot_thetar

```
real(dp), parameter, public mo_mpr_constants::pwp_matpot_thetar = 15000.0_dp
```

Definition at line 63 of file mo_mpr_constants.f90.

Referenced by mo_mpr_soilmoist::pwp().

17.26.2.19 vgenuchten_sandtresh

```
real(dp), parameter, public mo_mpr_constants::vgenuchten_sandtresh = 66.5_dp
```

Definition at line 40 of file mo_mpr_constants.f90.

Referenced by mo_mpr_soilmoist::genuchten().

17.26.2.20 vgenuchtenn_c1

```
real(dp), parameter, public mo_mpr_constants::vgenuchtenn_c1 = 1.392_dp
```

Definition at line 41 of file mo_mpr_constants.f90.

Referenced by mo_mpr_soilmoist::genuchten().

17.26.2.21 vgenuchtenn_c10

```
real(dp), parameter, public mo_mpr_constants::vgenuchtenn_c10 = -2.562_dp
```

Definition at line 50 of file mo_mpr_constants.f90.

Referenced by mo_mpr_soilmoist::genuchten().

17.26.2.22 vgenuchtenn_c11

```
real(dp), parameter, public mo_mpr_constants::vgenuchtenn_c11 = 7.0E-9_dp
```

Definition at line 51 of file mo_mpr_constants.f90.

Referenced by mo_mpr_soilmoist::genuchten().

17.26.2.23 vgenuchtenn_c12

```
real(dp), parameter, public mo_mpr_constants::vgenuchtenn_c12 = 4.004_dp
```

Definition at line 52 of file mo_mpr_constants.f90.

Referenced by mo_mpr_soilmoist::genuchten().

17.26.2.24 vgenuchtenn_c13

```
real(dp), parameter, public mo_mpr_constants::vgenuchtenn_c13 = 3.750_dp
```

Definition at line 53 of file mo_mpr_constants.f90.

Referenced by mo_mpr_soilmoist::genuchten().

17.26.2.25 vgenuchtenn_c14

```
real(dp), parameter, public mo_mpr_constants::vgenuchtenn_c14 = -0.016_dp
```

Definition at line 54 of file mo_mpr_constants.f90.

Referenced by mo_mpr_soilmoist::genuchten().

17.26.2.26 vgenuchtenn_c15

```
real(dp), parameter, public mo_mpr_constants::vgenuchtenn_c15 = -4.197_dp
```

Definition at line 55 of file mo_mpr_constants.f90.

Referenced by mo_mpr_soilmoist::genuchten().

17.26.2.27 vgenuchtenn_c16

```
real(dp), parameter, public mo_mpr_constants::vgenuchtenn_c16 = 0.013_dp
```

Definition at line 56 of file mo_mpr_constants.f90.

Referenced by mo_mpr_soilmoist::genuchten().

17.26.2.28 vgenuchtenn_c17

```
real(dp), parameter, public mo_mpr_constants::vgenuchtenn_c17 = 0.076_dp
```

Definition at line 57 of file mo_mpr_constants.f90.

Referenced by mo_mpr_soilmoist::genuchten().

17.26.2.29 vgenuchtenn_c18

```
real(dp), parameter, public mo_mpr_constants::vgenuchtenn_c18 = 0.276_dp
```

Definition at line 58 of file mo_mpr_constants.f90.

Referenced by mo_mpr_soilmoist::genuchten().

17.26.2.30 vgenuchtenn_c2

```
real(dp), parameter, public mo_mpr_constants::vgenuchtenn_c2 = 0.418_dp
```

Definition at line 42 of file mo_mpr_constants.f90.

Referenced by mo_mpr_soilmoist::genuchten().

17.26.2.31 vgenuchtenn_c3

```
real(dp), parameter, public mo_mpr_constants::vgenuchtenn_c3 = -0.024_dp
```

Definition at line 43 of file mo_mpr_constants.f90.

Referenced by mo_mpr_soilmoist::genuchten().

17.26.2.32 vgenuchtenn_c4

```
real(dp), parameter, public mo_mpr_constants::vgenuchtenn_c4 = 1.212_dp
```

Definition at line 44 of file mo_mpr_constants.f90.

Referenced by mo_mpr_soilmoist::genuchten().

17.26.2.33 vgenuchtenn_c5

```
real(dp), parameter, public mo_mpr_constants::vgenuchtenn_c5 = -0.704_dp
```

Definition at line 45 of file mo_mpr_constants.f90.

Referenced by mo_mpr_soilmoist::genuchten().

17.26.2.34 vgenuchtenn_c6

```
real(dp), parameter, public mo_mpr_constants::vgenuchtenn_c6 = -0.648_dp
```

Definition at line 46 of file mo_mpr_constants.f90.

Referenced by mo_mpr_soilmoist::genuchten().

17.26.2.35 vgenuchtenn_c7

```
real(dp), parameter, public mo_mpr_constants::vgenuchtenn_c7 = 0.023_dp
```

Definition at line 47 of file mo_mpr_constants.f90.

Referenced by mo_mpr_soilmoist::genuchten().

17.26.2.36 vgenuchtenn_c8

```
real(dp), parameter, public mo_mpr_constants::vgenuchtenn_c8 = 0.044_dp
```

Definition at line 48 of file mo_mpr_constants.f90.

Referenced by mo_mpr_soilmoist::genuchten().

17.26.2.37 vgenuchtenn_c9

```
real(dp), parameter, public mo_mpr_constants::vgenuchtenn_c9 = 3.168_dp
```

Definition at line 49 of file mo_mpr_constants.f90.

Referenced by mo_mpr_soilmoist::genuchten().

17.26.2.38 windmeasheight

```
real(dp), parameter, public mo_mpr_constants::windmeasheight = 10.0_dp
```

assumed meteorol. measurement hight for estimation of aeroResist and surfResist

Definition at line 66 of file mo_mpr_constants.f90.

Referenced by mo_multi_param_reg::aerodynamical_resistance().

17.27 mo_mpr_eval Module Reference

Runs MPR and writes to global effective parameters.

Functions/Subroutines

- subroutine, public mpr_eval (parameterset, opti_domain_indices)

  Runs MPR and writes to global effective parameters.

17.27.1 Detailed Description

Runs MPR and writes to global effective parameters.

Runs MPR and writes to global effective parameters

Authors

Robert Schweppe

Date

Feb 2018

17.27.2 Function/Subroutine Documentation

17.27.2.1 mpr_eval()

```
subroutine, public mo_mpr_eval::mpr_eval (
             real(dp), dimension(:), intent(in), optional parameterset,
             integer(i4), dimension(:), intent(in), optional opti_domain_indices )
```

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

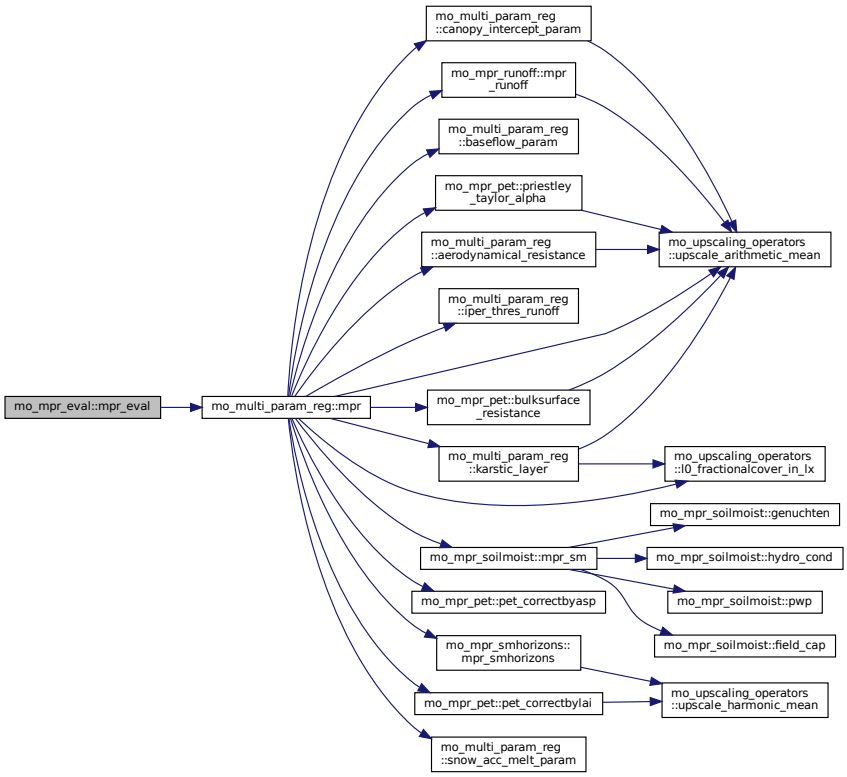
Feb 2013

Definition at line 71 of file mo_mpr_eval.f90.

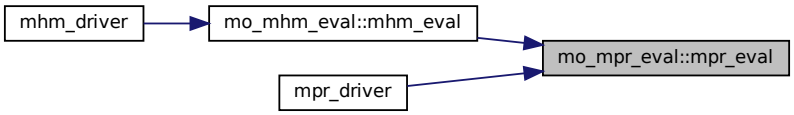
References mo_common_variables::domainmeta, mo_mpr_global_variables::l0_asp, mo_mpr_global_variables↩ ::l0_geounit, mo_mpr_global_variables::l0_gridded_lai, mo_common_variables::l0_l1_remap, mo_common_↩ variables::l0_lcover, mo_mpr_global_variables::l0_slope_emp, mo_mpr_global_variables::l0_soilid, mo_mpr↩ _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, and mo_multi_param_reg::mpr().

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:



17.28 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.

17.28.1 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

17.28.2 Variable Documentation

17.28.2.1 file_aspect

```
character(len = *), parameter mo_mpr_file::file_aspect = 'aspect.asc'
```

aspect input data file

Definition at line 44 of file mo_mpr_file.f90.

Referenced by mo_read_wrapper::read_data().

17.28.2.2 file_geolut

```
character(len = *), parameter mo_mpr_file::file_geolut = 'geology_classdefinition.txt'
```

geological formation lookup table file

Definition at line 61 of file mo_mpr_file.f90.

Referenced by mo_read_wrapper::read_data().

17.28.2.3 file_hydrogeoclass

```
character(len = *), parameter mo_mpr_file::file_hydrogeoclass = 'geology_class.asc'
```

hydrogeological classes input data file

Definition at line 48 of file mo_mpr_file.f90.

Referenced by mo_startup::constants_init(), and mo_read_wrapper::read_data().

17.28.2.4 file_laiclass

```
character(len = *), parameter mo_mpr_file::file_laiclass = 'LAI_class.asc'
```

LAI classes input data file.

Definition at line 56 of file mo_mpr_file.f90.

Referenced by mo_read_wrapper::read_data().

17.28.2.5 file_lailut

```
character(len = *), parameter mo_mpr_file::file_lailut = 'LAI_classdefinition.txt'
```

LAI classes lookup table file.

Definition at line 66 of file mo_mpr_file.f90.

Referenced by mo_read_wrapper::read_data().

17.28.2.6 file_main

```
character(len = *), parameter mo_mpr_file::file_main = 'mpr_driver.f90'
```

Driver file.

Definition at line 22 of file mo_mpr_file.f90.

17.28.2.7 file_meteo_binary_end

```
character(len = *), parameter mo_mpr_file::file_meteo_binary_end = '.bin'
```

File ending of meteo files.

Definition at line 75 of file mo_mpr_file.f90.

17.28.2.8 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.

Definition at line 71 of file mo_mpr_file.f90.

Referenced by mo_startup::l2_variable_init().

17.28.2.9 file_namelist_mpr

```
character(len = *), parameter mo_mpr_file::file_namelist_mpr = 'mpr.nml'
```

Namelist file name.

Definition at line 24 of file mo_mpr_file.f90.

17.28.2.10 file_namelist_mpr_param

```
character(len = *), parameter mo_mpr_file::file_namelist_mpr_param = 'mpr_parameter.nml'
```

Parameter namelists file name.

Definition at line 28 of file mo_mpr_file.f90.

Referenced by mpr_driver().

17.28.2.11 file_slope

```
character(len = *), parameter mo_mpr_file::file_slope = 'slope.asc'
```

slope input data file

Definition at line 40 of file mo_mpr_file.f90.

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

17.28.2.12 file_soil_database

```
character(len = *), parameter mo_mpr_file::file_soil_database = 'soil_classdefinition.txt'
```

Soil database file (iFlag_soilDB = 0) = classical mHM format.

Definition at line 34 of file mo_mpr_file.f90.

Referenced by mo_read_wrapper::read_data().

17.28.2.13 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)

Definition at line 36 of file mo_mpr_file.f90.

Referenced by mo_read_wrapper::read_data().

17.28.2.14 file_soilclass

```
character(len = *), parameter mo_mpr_file::file_soilclass = 'soil_class.asc'
```

soil classes input data file

Definition at line 52 of file mo_mpr_file.f90.

Referenced by mo_read_wrapper::read_data().

17.28.2.15 uaspect

```
integer, parameter mo_mpr_file::uaspect = 55
```

Unit for aspect input data file.

Definition at line 46 of file mo_mpr_file.f90.

Referenced by mo_read_wrapper::read_data().

17.28.2.16 ugeolut

```
integer, parameter mo_mpr_file::ugeolut = 64
```

Unit for geological formation lookup table file.

Definition at line 63 of file mo_mpr_file.f90.

Referenced by mo_read_wrapper::read_data().

17.28.2.17 uhydrogeoclass

```
integer, parameter mo_mpr_file::uhydrogeoclass = 58
```

Unit for hydrogeological classes input data file.

Definition at line 50 of file mo_mpr_file.f90.

Referenced by mo_read_wrapper::read_data().

17.28.2.18 ulaiclass

```
integer, parameter mo_mpr_file::ulaiclass = 60
```

Unit for LAI input data file.

Definition at line 58 of file mo_mpr_file.f90.

Referenced by mo_read_wrapper::read_data().

17.28.2.19 ulailut

```
integer, parameter mo_mpr_file::ulailut = 65
```

Unit for LAI classes lookup table file.

Definition at line 68 of file mo_mpr_file.f90.

Referenced by mo_read_wrapper::read_data().

17.28.2.20 umeteo

```
integer, parameter mo_mpr_file::umeteo = 51
```

Unit for meteo files.

Definition at line 77 of file mo_mpr_file.f90.

17.28.2.21 umeteo_header

```
integer, parameter mo_mpr_file::umeteo_header = 50
```

Unit for meteo header file.

Definition at line 73 of file mo_mpr_file.f90.

Referenced by mo_startup::l2_variable_init().

17.28.2.22 unamelist_mpr

```
integer, parameter mo_mpr_file::unamelist_mpr = 80
```

Unit for namelist.

Definition at line 26 of file mo_mpr_file.f90.

17.28.2.23 unamelist_mpr_param

```
integer, parameter mo_mpr_file::unamelist_mpr_param = 31
```

Unit for namelist.

Definition at line 30 of file mo_mpr_file.f90.

Referenced by mpr_driver().

17.28.2.24 uslope

```
integer, parameter mo_mpr_file::uslope = 54
```

Unit for slope input data file.

Definition at line 42 of file mo_mpr_file.f90.

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

17.28.2.25 usoil_database

```
integer, parameter mo_mpr_file::usoil_database = 52
```

Unit for soil data base.

Definition at line 38 of file mo_mpr_file.f90.

Referenced by mo_soil_database::read_soil_lut().

17.28.2.26 usoilclass

```
integer, parameter mo_mpr_file::usoilclass = 59
```

Unit for soil classes input data file.

Definition at line 54 of file mo_mpr_file.f90.

Referenced by mo_read_wrapper::read_data().

17.28.2.27 version

```
character(len = *), parameter mo_mpr_file::version = '0.1'
```

Current mHM model version.

Definition at line 18 of file mo_mpr_file.f90.

17.28.2.28 version_date

```
character(len = *), parameter mo_mpr_file::version_date = 'Jun 2019'
```

Time of current mHM model version release.

Definition at line 20 of file mo_mpr_file.f90.

17.29 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
- real(dp), dimension(:), allocatable, public laiboundaries
- 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

17.29.1 Detailed Description

Global variables for mpr only.

TODO: add description

Authors

Robert Schweppe

Date

Dec 2017

17.29.2 Variable Documentation

17.29.2.1 dirgridded_lai

```
character(256), dimension(:), allocatable, public mo_mpr_global_variables::dirgridded_lai
```

Definition at line 115 of file mo_mpr_global_variables.f90.

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().

17.29.2.2 fracsealed_cityarea

```
real(dp), public mo_mpr_global_variables::fracsealed_cityarea
```

Definition at line 92 of file mo_mpr_global_variables.f90.

Referenced by mo_multi_param_reg::mpr(), and mo_mpr_read_config::mpr_read_config().

17.29.2.3 geounitkar

```
integer(i4), dimension(:), allocatable, public mo_mpr_global_variables::geounitkar
```

Definition at line 76 of file mo_mpr_global_variables.f90.

Referenced by mo_multi_param_reg::karstic_layer(), and mo_read_wrapper::read_data().

17.29.2.4 geounitlist

```
integer(i4), dimension(:), allocatable, public mo_mpr_global_variables::geounitlist
```

Definition at line 75 of file mo_mpr_global_variables.f90.

Referenced by mo_multi_param_reg::baseflow_param(), mo_startup::constants_init(), mo_multi_param_reg↩::karstic_layer(), and mo_read_wrapper::read_data().

17.29.2.5 horizondepth_mhm

```
real(dp), dimension(:), allocatable, public mo_mpr_global_variables::horizondepth_mhm
```

Definition at line 31 of file mo_mpr_global_variables.f90.

Referenced by mo_common_mhm_mrm_restart::check_dimension_consistency(), 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(), mo_init_states::variables_default_init(), mo_mpr_restart::write_mpr_restart_files(), and mo_restart::write_restart_files().

17.29.2.6 iflag_soildb

```
integer(i4), public mo_mpr_global_variables::iflag_soildb
```

Definition at line 29 of file mo_mpr_global_variables.f90.

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().

17.29.2.7 inputformat_gridded_lai

```
character(256), public mo_mpr_global_variables::inputformat_gridded_lai
```

Definition at line 81 of file mo_mpr_global_variables.f90.

Referenced by mo_mpr_read_config::mpr_read_config(), and mo_prepare_gridded_lai::prepare_gridded_daily_lai_data().

17.29.2.8 l0_asp

```
real(dp), dimension(:), allocatable, public mo_mpr_global_variables::l0_asp
```

Definition at line 106 of file mo_mpr_global_variables.f90.

Referenced by mo_mpr_startup::l0_check_input(), mo_mpr_eval::mpr_eval(), and mo_read_wrapper::read_data().

17.29.2.9 l0_geounit

```
integer(i4), dimension(:), allocatable, public mo_mpr_global_variables::l0_geounit
```

Definition at line 109 of file mo_mpr_global_variables.f90.

Referenced by mo_mpr_startup::l0_check_input(), mo_mpr_eval::mpr_eval(), and mo_read_wrapper::read_data().

17.29.2.10 l0_gridded_lai

```
real(dp), dimension(:, :), allocatable, public mo_mpr_global_variables::l0_gridded_lai
```

Definition at line 102 of file mo_mpr_global_variables.f90.

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().

17.29.2.11 l0_slope

```
real(dp), dimension(:), allocatable, public mo_mpr_global_variables::l0_slope
```

Definition at line 105 of file mo_mpr_global_variables.f90.

Referenced by mo_mpr_startup::l0_check_input(), mo_mpr_startup::l0_variable_init(), mo_mrm_net_startup::l11_calc_celerity(), mo_mrm_restart::mrm_read_restart_config(), mo_mrm_restart::mrm_write_restart(), and mo_read_wrapper::read_data().

17.29.2.12 l0_slope_emp

```
real(dp), dimension(:), allocatable, public mo_mpr_global_variables::l0_slope_emp
```

Definition at line 100 of file mo_mpr_global_variables.f90.

Referenced by mo_mpr_startup::l0_variable_init(), and mo_mpr_eval::mpr_eval().

17.29.2.13 l0_soilid

```
integer(i4), dimension(:, :), allocatable, public mo_mpr_global_variables::l0_soilid
```

Definition at line 108 of file mo_mpr_global_variables.f90.

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().

17.29.2.14 l1_aeroresist

```
real(dp), dimension(:, :, :), allocatable, public mo_mpr_global_variables::l1_aeroresist
```

Definition at line 137 of file mo_mpr_global_variables.f90.

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().

17.29.2.15 l1_alpha

```
real(dp), dimension(:, :, :), allocatable, public mo_mpr_global_variables::l1_alpha
```

Definition at line 124 of file mo_mpr_global_variables.f90.

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().

17.29.2.16 l1_degday

```
real(dp), dimension(:, :, :), allocatable, public mo_mpr_global_variables::l1_degday
```

Definition at line 130 of file mo_mpr_global_variables.f90.

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().

17.29.2.17 l1_degdayinc

```
real(dp), dimension(:, :, :), allocatable, public mo_mpr_global_variables::l1_degdayinc
```

Definition at line 125 of file mo_mpr_global_variables.f90.

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().

17.29.2.18 l1_degdaymax

```
real(dp), dimension(:, :, :), allocatable, public mo_mpr_global_variables::l1_degdaymax
```

Definition at line 127 of file mo_mpr_global_variables.f90.

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().

17.29.2.19 l1_degdaynopre

```
real(dp), dimension(:, :, :), allocatable, public mo_mpr_global_variables::l1_degdaynopre
```

Definition at line 128 of file mo_mpr_global_variables.f90.

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().

17.29.2.20 l1_fasp

```
real(dp), dimension(:, :, :), allocatable, public mo_mpr_global_variables::l1_fasp
```

Definition at line 132 of file mo_mpr_global_variables.f90.

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().

17.29.2.21 l1_froots

```
real(dp), dimension(:, :, :), allocatable, public mo_mpr_global_variables::l1_froots
```

Definition at line 139 of file mo_mpr_global_variables.f90.

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().

17.29.2.22 l1_fsealed

```
real(dp), dimension(:, :, :), allocatable, public mo_mpr_global_variables::l1_fsealed
```

Definition at line 122 of file mo_mpr_global_variables.f90.

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().

17.29.2.23 l1_harsamcoeff

```
real(dp), dimension(:, :, :), allocatable, public mo_mpr_global_variables::l1_harsamcoeff
```

Definition at line 135 of file mo_mpr_global_variables.f90.

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().

17.29.2.24 l1_jarvis_thresh_c1

```
real(dp), dimension(:, :, :), allocatable, public mo_mpr_global_variables::l1_jarvis_thresh_c1
```

Definition at line 151 of file mo_mpr_global_variables.f90.

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().

17.29.2.25 l1_karstloss

```
real(dp), dimension(:, :, :), allocatable, public mo_mpr_global_variables::l1_karstloss
```

Definition at line 131 of file mo_mpr_global_variables.f90.

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().

17.29.2.26 l1_kbaseflow

```
real(dp), dimension(:, :, :), allocatable, public mo_mpr_global_variables::l1_kbaseflow
```

Definition at line 144 of file mo_mpr_global_variables.f90.

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().

17.29.2.27 l1_kfastflow

```
real(dp), dimension(:, :, :), allocatable, public mo_mpr_global_variables::l1_kfastflow
```

Definition at line 142 of file mo_mpr_global_variables.f90.

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().

17.29.2.28 l1_kperco

```
real(dp), dimension(:, :, :), allocatable, public mo_mpr_global_variables::l1_kperco
```

Definition at line 145 of file mo_mpr_global_variables.f90.

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().

17.29.2.29 l1_kslowflow

```
real(dp), dimension(:, :, :), allocatable, public mo_mpr_global_variables::l1_kslowflow
```

Definition at line 143 of file mo_mpr_global_variables.f90.

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().

17.29.2.30 l1_maxinter

```
real(dp), dimension(:, :, :), allocatable, public mo_mpr_global_variables::l1_maxinter
```

Definition at line 140 of file mo_mpr_global_variables.f90.

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().

17.29.2.31 l1_petlaicorfactor

```
real(dp), dimension(:, :, :), allocatable, public mo_mpr_global_variables::l1_petlaicorfactor
```

Definition at line 133 of file mo_mpr_global_variables.f90.

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().

17.29.2.32 l1_prietayalpha

```
real(dp), dimension(:, :, :), allocatable, public mo_mpr_global_variables::l1_prietayalpha
```

Definition at line 136 of file mo_mpr_global_variables.f90.

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().

17.29.2.33 l1_sealedthresh

```
real(dp), dimension(:, :, :), allocatable, public mo_mpr_global_variables::l1_sealedthresh
```

Definition at line 155 of file mo_mpr_global_variables.f90.

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().

17.29.2.34 l1_soilmoistexp

```
real(dp), dimension(:, :, :), allocatable, public mo_mpr_global_variables::l1_soilmoistexp
```

Definition at line 149 of file mo_mpr_global_variables.f90.

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().

17.29.2.35 l1_soilmoistfc

```
real(dp), dimension(:, :, :), allocatable, public mo_mpr_global_variables::l1_soilmoistfc
```

Definition at line 146 of file mo_mpr_global_variables.f90.

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().

17.29.2.36 l1_soilmoistsat

```
real(dp), dimension(:, :, :), allocatable, public mo_mpr_global_variables::l1_soilmoistsat
```

Definition at line 148 of file mo_mpr_global_variables.f90.

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().

17.29.2.37 l1_surfresist

```
real(dp), dimension(:, :, :), allocatable, public mo_mpr_global_variables::l1_surfresist
```

Definition at line 138 of file mo_mpr_global_variables.f90.

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().

17.29.2.38 l1_tempthresh

```
real(dp), dimension(:, :, :), allocatable, public mo_mpr_global_variables::l1_tempthresh
```

Definition at line 153 of file mo_mpr_global_variables.f90.

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().

17.29.2.39 l1_unsatthresh

```
real(dp), dimension(:, :, :), allocatable, public mo_mpr_global_variables::l1_unsatthresh
```

Definition at line 154 of file mo_mpr_global_variables.f90.

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().

17.29.2.40 l1_wiltingpoint

```
real(dp), dimension(:, :, :), allocatable, public mo_mpr_global_variables::l1_wiltingpoint
```

Definition at line 157 of file mo_mpr_global_variables.f90.

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().

17.29.2.41 laiboundaries

```
real(dp), dimension(:), allocatable, public mo_mpr_global_variables::laiboundaries
```

Definition at line 87 of file mo_mpr_global_variables.f90.

Referenced by mo_common_mhm_mrm_restart::check_dimension_consistency(), mo_prepare_gridded_lai↩ ::prepare_gridded_daily_lai_data(), mo_prepare_gridded_lai::prepare_gridded_mean_monthly_lai_data(), mo↩ _read_wrapper::read_data(), and mo_restart::write_restart_files().

17.29.2.42 lailut

```
real(dp), dimension(:, :), allocatable, public mo_mpr_global_variables::lailut
```

Definition at line 89 of file mo_mpr_global_variables.f90.

Referenced by mo_read_wrapper::read_data().

17.29.2.43 laiper

```
type(period), dimension(:), allocatable, public mo_mpr_global_variables::laiper
```

Definition at line 91 of file mo_mpr_global_variables.f90.

17.29.2.44 laiunitlist

```
integer(i4), dimension(:), allocatable, public mo_mpr_global_variables::laiunitlist
```

Definition at line 88 of file mo_mpr_global_variables.f90.

Referenced by mo_read_wrapper::read_data().

17.29.2.45 ngeounits

```
integer(i4), public mo_mpr_global_variables::ngeounits
```

Definition at line 74 of file mo_mpr_global_variables.f90.

Referenced by mo_mpr_read_config::mpr_read_config(), and mo_read_wrapper::read_data().

17.29.2.46 nlai

```
integer(i4), public mo_mpr_global_variables::nlai
```

Definition at line 86 of file mo_mpr_global_variables.f90.

Referenced by mo_common_mhm_mrm_restart::check_dimension_consistency(), 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().

17.29.2.47 nlaiclass

```
integer(i4), public mo_mpr_global_variables::nlaiclass
```

Definition at line 85 of file mo_mpr_global_variables.f90.

Referenced by mo_read_wrapper::read_data().

17.29.2.48 nsoilhorizons_mhm

```
integer(i4), public mo_mpr_global_variables::nsoilhorizons_mhm
```

Definition at line 30 of file mo_mpr_global_variables.f90.

Referenced by mo_common_mhm_mrm_restart::check_dimension_consistency(), 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().

17.29.2.49 nsoiltypes

```
integer(i4), public mo_mpr_global_variables::nsoiltypes
```

Definition at line 28 of file mo_mpr_global_variables.f90.

Referenced by mo_soil_database::generate_soil_database(), mo_mpr_startup::l0_variable_init(), and mo_soil_↩database::read_soil_lut().

17.29.2.50 soildb

```
type(soiltype), public mo_mpr_global_variables::soildb
```

Definition at line 69 of file mo_mpr_global_variables.f90.

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().

17.29.2.51 tillagedepth

```
real(dp), public mo_mpr_global_variables::tillagedepth
```

Definition at line 26 of file mo_mpr_global_variables.f90.

Referenced by mo_mpr_read_config::mpr_read_config(), and mo_soil_database::read_soil_lut().

17.29.2.52 timestep_lai_input

```
integer(i4), public mo_mpr_global_variables::timestep_lai_input
```

Definition at line 82 of file mo_mpr_global_variables.f90.

Referenced by mo_common_datetime_type::datetimeinfo_update_lai_timestep(), mo_mpr_startup::l0_check_↩ input(), mo_mpr_read_config::mpr_read_config(), mo_prepare_gridded_lai::prepare_gridded_daily_lai_data(), and mo_read_wrapper::read_data().

17.30 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.

17.30.1 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

17.30.2 Function/Subroutine Documentation

17.30.2.1 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
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

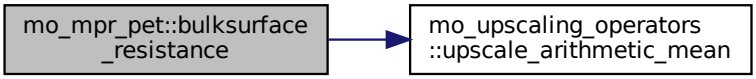
Apr 2013

Definition at line 408 of file mo_mpr_pet.f90.

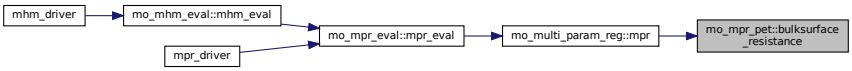
References mo_mpr_constants::lai_factor_surfresi, mo_mpr_constants::lai_offset_surfresi, mo_mpr_constants::max_surfresist, and mo_upscaling_operators::upscale_arithmetic_mean().

Referenced by mo_multi_param_reg::mpr().

Here is the call graph for this function:



Here is the caller graph for this function:



17.30.2.2 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

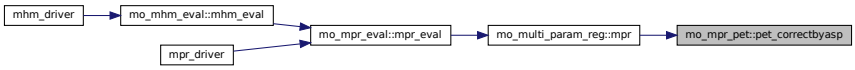
Date

Dec 2012

Definition at line 215 of file mo_mpr_pet.f90.

Referenced by mo_multi_param_reg::mpr().

Here is the caller graph for this function:



17.30.2.3 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

Parameters

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

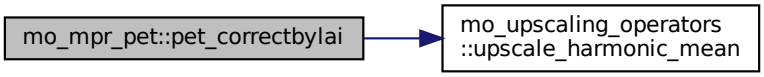
May. 2017

Definition at line 81 of file mo_mpr_pet.f90.

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:

17.30.2.4 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

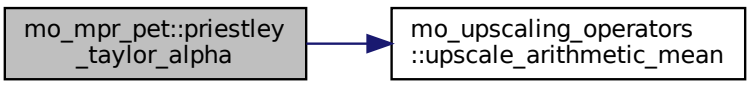
Apr 2013

Definition at line 312 of file mo_mpr_pet.f90.

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:

17.31 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.

17.31.1 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

17.31.2 Function/Subroutine Documentation

17.31.2.1 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

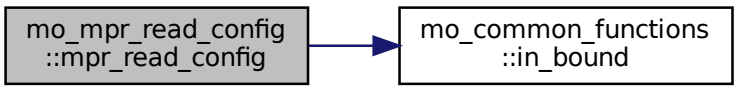
Aug 2015

Definition at line 54 of file mo_mpr_read_config.f90.

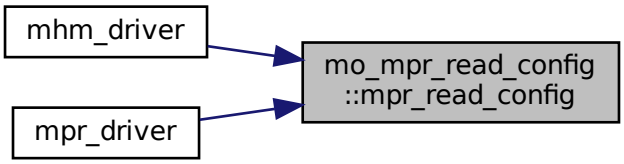
References mo_mpr_global_variables::dirgridded_lai, mo_common_variables::domainmeta, mo_common_↩ constants::eps_dp, mo_mpr_global_variables::fracsealed_cityarea, mo_common_variables::global_parameters, mo_common_variables::global_parameters_name, mo_mpr_global_variables::horizondepth_mhm, mo_mpr↩ _global_variables::iflag_soildb, mo_common_functions::in_bound(), mo_mpr_global_variables::inputformat_↩ gridded_lai, mo_mpr_constants::maxgeounit, mo_common_constants::maxnodomains, mo_mpr_constants↩ ::maxnosoilhorizons, mo_common_constants::ncolpars, mo_mpr_global_variables::ngeounits, mo_common_↩ constants::nodata_dp, mo_mpr_global_variables::nsoilhorizons_mhm, 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:



17.32 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 (OutFile)

  write restart files for each domain
- 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)

17.32.1 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

17.32.2 Function/Subroutine Documentation

17.32.2.1 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 )
```

Definition at line 413 of file mo_mpr_restart.f90.

17.32.2.2 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 )
```

Definition at line 368 of file mo_mpr_restart.f90.

17.32.2.3 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 )
```

Definition at line 458 of file mo_mpr_restart.f90.

17.32.2.4 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 )
```

Definition at line 515 of file mo_mpr_restart.f90.

17.32.2.5 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*	

Parameters

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

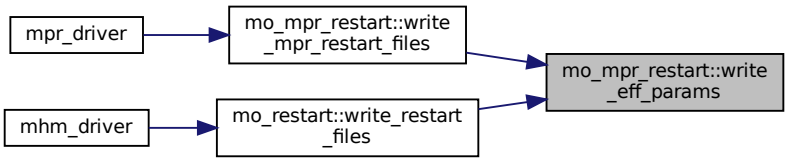
Jun 2018

Definition at line 209 of file mo_mpr_restart.f90.

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_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:



17.32.2.6 write_mpr_restart_files()

```
subroutine, public mo_mpr_restart::write_mpr_restart_files (
             character(256), dimension(:), intent(in) OutFile )
```

write restart files for each domain

write restart files for each domain. For each domain three restart files are written. These are xxx_states.nc, xxx_↩ L11_config.nc, and xxx_config.nc (xxx being the three digit domain 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(:) :: OutFile*	Output Path for each domain

Authors

Stephan Thober

Date

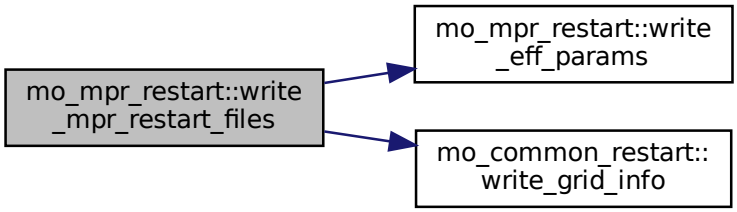
Jun 2014

Definition at line 98 of file mo_mpr_restart.f90.

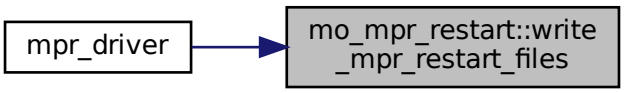
References mo_common_variables::domainmeta, mo_mpr_global_variables::horizondepth_mhm, mo_common_constants::laivarname, mo_common_constants::landcoverperiodsvarname, mo_common_variables::lc_year_end, mo_common_variables::lc_year_start, mo_common_variables::level1, mo_mpr_global_variables::nlai, mo_common_variables::nlcoverscene, mo_mpr_global_variables::nsoilhorizons_mhm, mo_common_constants::soilhorizonsvarname, 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:



17.33 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

17.33.1 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

17.33.2 Function/Subroutine Documentation

17.33.2.1 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)
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

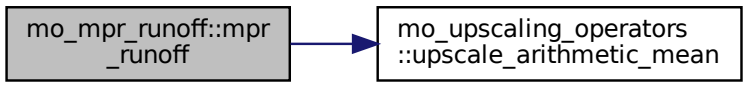
Dec 2012

Definition at line 75 of file mo_mpr_runoff.f90.

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:

17.34 mo_mpr_smhorizons Module Reference

setting up the soil moisture horizons

Functions/Subroutines

- subroutine, public 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

17.34.1 Detailed Description

setting up the soil moisture horizons

This module sets up the soil moisture horizons

Authors

Stephan Thober, Rohini Kumar

Date

Dec 2012

17.34.2 Function/Subroutine Documentation

17.34.2.1 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

Parameters

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
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

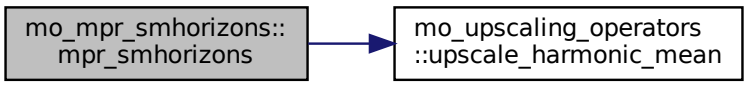
in this case the second dimension of soilId0 = 1

Definition at line 121 of file mo_mpr_smhorizons.f90.

References 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:

17.35 mo_mpr_soilmoist Module Reference

Multiscale parameter regionalization (MPR) for soil moisture.

Functions/Subroutines

- subroutine, public mpr_sm (param, processMatrix, 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 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

17.35.1 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

17.35.2 Function/Subroutine Documentation

17.35.2.1 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

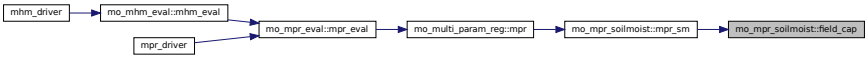
Dec 2012

Definition at line 564 of file mo_mpr_soilmoist.f90.

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:



17.35.2.2 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

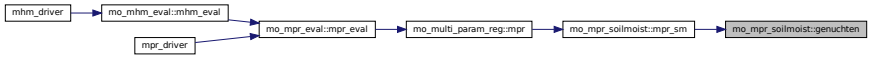
Dec 2012

Definition at line 630 of file mo_mpr_soilmoist.f90.

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:



17.35.2.3 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

Definition at line 740 of file mo_mpr_soilmoist.f90.

References mo_mpr_constants::ks_c.

Referenced by mpr_sm().

Here is the caller graph for this function:

17.35.2.4 mpr_sm()

```
subroutine, public mo_mpr_soilmoist::mpr_sm (
             real(dp), dimension(13), intent(in) param,
             integer(i4), dimension(:, :), intent(in) processMatrix,
             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

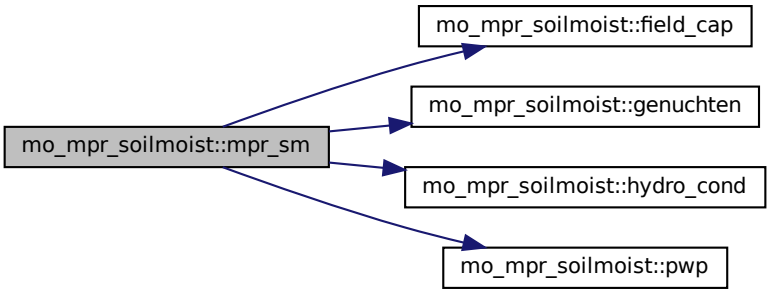
SOIL TYPE

Definition at line 100 of file mo_mpr_soilmoist.f90.

References mo_mpr_constants::bulkdens_orgmatter, field_cap(), genuchten(), hydro_cond(), mo_mpr_global_↩ variables::iflag_soildb, 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:

17.35.2.5 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

Parameters

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

Definition at line 493 of file mo_mpr_soilmoist.f90.

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:

17.36 mo_mpr_startup Module Reference

Startup procedures for mHM.

Functions/Subroutines

- subroutine, public mpr_initialize

  Initialize main mHM variables.
- subroutine l0_check_input (iDomain)

  Check for errors in L0 input data.
- subroutine l0_variable_init (iDomain)

  level 0 variable initialization
- subroutine, public init_eff_params (ncells1)

  Allocation of space for mHM related L1 and L11 variables.

17.36.1 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

17.36.2 Function/Subroutine Documentation

17.36.2.1 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 domain. 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

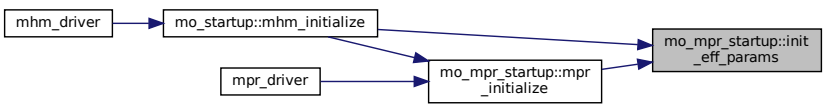
Jan 2013

Definition at line 401 of file mo_mpr_startup.f90.

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, and mo_common_constants::p1_initstatefluxes.

Referenced by mo_startup::mhm_initialize(), and mpr_initialize().

Here is the caller graph for this function:



17.36.2.2 l0_check_input()

```
subroutine mo_mpr_startup::l0_check_input (
            integer(i4), intent(in) iDomain )
```

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) :: iDomain*	domain id

Authors

Rohini Kumar

Date

Jan 2013

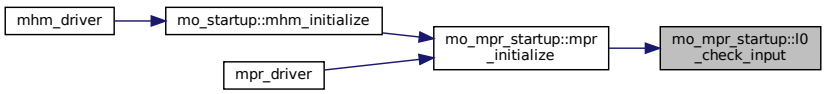
by default; when iFlag_soilDB = 0

Definition at line 143 of file mo_mpr_startup.f90.

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_common_variables::nlcoverscene, mo_mpr_global_variables::nsoilhorizons_mhm, and mo_mpr_global_variables::timestep_lai_input.

Referenced by mpr_initialize().

Here is the caller graph for this function:



17.36.2.3 l0_variable_init()

```
subroutine mo_mpr_startup::l0_variable_init (
            integer(i4), intent(in) iDomain )
```

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) :: iDomain*	domain id

Authors

Rohini Kumar

Date

Jan 2013

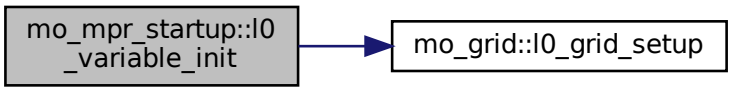
by default; when iFlag_soilDB = 0

Definition at line 270 of file mo_mpr_startup.f90.

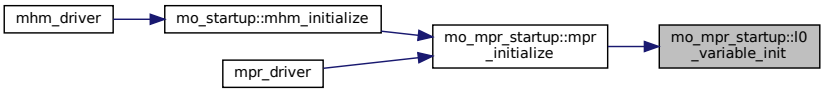
References mo_mpr_global_variables::iflag_soildb, mo_grid::l0_grid_setup(), mo_mpr_global_variables::l0_slope, mo_mpr_global_variables::l0_slope_emp, mo_mpr_global_variables::l0_soilid, mo_common_variables::level0, mo_mpr_global_variables::nsoilhorizons_mhm, mo_mpr_global_variables::nsoiltypes, and mo_mpr_global_↩ variables::soildb.

Referenced by mpr_initialize().

Here is the call graph for this function:



Here is the caller graph for this function:



17.36.2.4 mpr_initialize()

```
subroutine, public mo_mpr_startup::mpr_initialize
```

Initialize main mHM variables.

Initialize main mHM variables for a given domain. 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

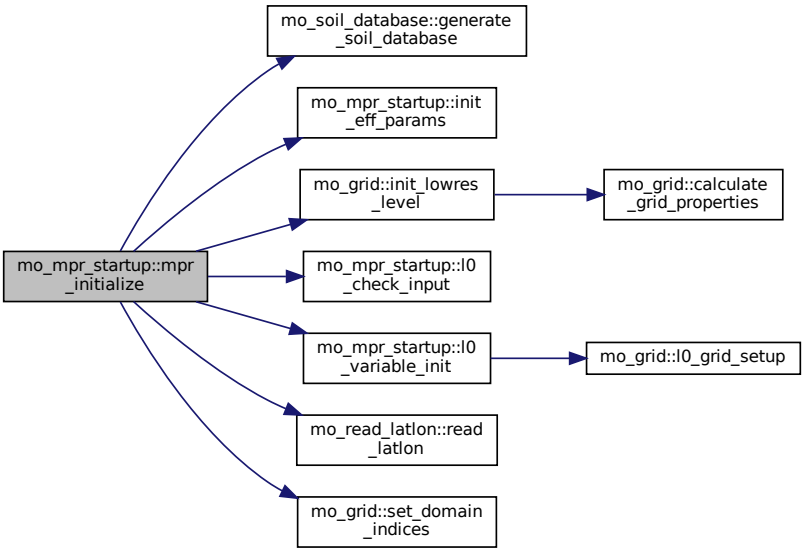
Dec 2012

Definition at line 71 of file mo_mpr_startup.f90.

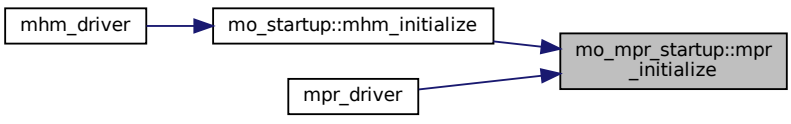
References mo_common_variables::domainmeta, mo_soil_database::generate_soil_database(), init_eff_↩ params(), mo_grid::init_lowres_level(), l0_check_input(), mo_common_variables::l0_l1_remap, l0_variable_init(), mo_common_variables::level0, mo_common_variables::level1, mo_read_latlon::read_latlon(), mo_common_↩ variables::resolutionhydrology, and mo_grid::set_domain_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:



17.37 mo_mrm_constants Module Reference

Provides mRM specific constants.

Variables

- integer(i4), parameter, public noutflxstate = 2_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/)

17.37.1 Detailed Description

Provides mRM specific constants.

Provides mRM specific constants such as flood plain elevation.

Authors

Stephan Thober

Date

Aug 2015

17.37.2 Variable Documentation

17.37.2.1 deltah

```
real(dp), parameter, public mo_mrm_constants::deltah = 5.000_dp
```

Definition at line 37 of file mo_mrm_constants.f90.

Referenced by mo_mrm_net_startup::moveup().

17.37.2.2 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/)
```

Definition at line 41 of file mo_mrm_constants.f90.

Referenced by mo_mrm_mpr::mrm_init_param(), and mo_mrm_mpr::mrm_update_param().

17.37.2.3 maxnogauges

```
integer(i4), parameter, public mo_mrm_constants::maxnogauges = 50_i4
```

Definition at line 27 of file mo_mrm_constants.f90.

Referenced by mo_mrm_read_config::mrm_read_config().

17.37.2.4 noutflxstate

```
integer(i4), parameter, public mo_mrm_constants::noutflxstate = 2_i4
```

Definition at line 18 of file mo_mrm_constants.f90.

17.37.2.5 nroutingstates

```
integer(i4), parameter, public mo_mrm_constants::nroutingstates = 2
```

Definition at line 20 of file mo_mrm_constants.f90.

Referenced by mo_mrm_riv_temp_class::init(), mo_mrm_restart::mrm_read_restart_states(), mo_mrm_restart↩ ::mrm_write_restart(), and mo_mrm_init::variables_alloc_routing().

17.37.2.6 rout_space_weight

```
real(dp), parameter, public mo_mrm_constants::rout_space_weight = 0._dp
```

Definition at line 32 of file mo_mrm_constants.f90.

Referenced by mo_mrm_mpr::mrm_update_param().

17.38 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

17.38.1 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

17.38.2 Variable Documentation

17.38.2.1 file_config

```
character(len = *), parameter mo_mrm_file::file_config = 'ConfigFile.log'
```

file defining mHM's outputs

Definition at line 58 of file mo_mrm_file.f90.

17.38.2.2 file_daily_discharge

```
character(len = *), parameter mo_mrm_file::file_daily_discharge = 'daily_discharge.out'
```

file defining optimazation outputs

Definition at line 63 of file mo_mrm_file.f90.

Referenced by mo_mrm_write::write_daily_obs_sim_discharge().

17.38.2.3 file_defoutput

```
character(len = *), parameter mo_mrm_file::file_defoutput = 'mrm_outputs.nml'
```

file defining mRM's outputs

Definition at line 53 of file mo_mrm_file.f90.

Referenced by mo_mrm_init::config_output(), mo_mrm_read_config::mrm_read_config(), and mo_mrm_init::print↩
_startup_message().

17.38.2.4 file_facc

```
character(len = *), parameter mo_mrm_file::file_facc = 'facc.asc'
```

Definition at line 32 of file mo_mrm_file.f90.

Referenced by mo_mrm_read_data::mrm_read_l0_data().

17.38.2.5 file_fdir

```
character(len = *), parameter mo_mrm_file::file_fdir = 'fdir.asc'
```

flow direction input data file

Definition at line 36 of file mo_mrm_file.f90.

Referenced by mo_mrm_read_data::mrm_read_l0_data().

17.38.2.6 file_gaugeloc

```
character(len = *), parameter mo_mrm_file::file_gaugeloc = 'idgauges.asc'
```

gauge location input data file

Definition at line 45 of file mo_mrm_file.f90.

Referenced by mo_mrm_read_data::mrm_read_l0_data().

17.38.2.7 file_gw_output

```
character(len = *), parameter mo_mrm_file::file_gw_output = 'mRM_gw_Fluxes_States.nc'
```

file containing mrm output for groundwater coupling

Definition at line 73 of file mo_mrm_file.f90.

17.38.2.8 file_main

```
character(len = *), parameter mo_mrm_file::file_main = 'mrm_driver.f90'
```

Driver file.

Definition at line 22 of file mo_mrm_file.f90.

Referenced by mo_mrm_init::print_startup_message().

17.38.2.9 file_mrm_output

```
character(len = *), parameter mo_mrm_file::file_mrm_output = 'mRM_Fluxes_States.nc'
```

file containing mrm output

Definition at line 70 of file mo_mrm_file.f90.

Referenced by mo_mrm_write_fluxes_states::createoutputfile().

17.38.2.10 file_namelist_mrm

```
character(len = *), parameter mo_mrm_file::file_namelist_mrm = 'mrm.nml'
```

Namelist file name.

Definition at line 24 of file mo_mrm_file.f90.

Referenced by mo_mrm_init::config_output().

17.38.2.11 file_namelist_param_mrm

```
character(len = *), parameter mo_mrm_file::file_namelist_param_mrm = 'mrm_parameter.nml'
```

Parameter namelists file name.

Definition at line 28 of file mo_mrm_file.f90.

Referenced by mo_mrm_init::config_output().

17.38.2.12 file_slope

```
character(len=*), parameter mo_mrm_file::file_slope = 'slope.asc'
```

flow direction input data file

Definition at line 40 of file mo_mrm_file.f90.

17.38.2.13 ncfile_discharge

```
character(len = *), parameter mo_mrm_file::ncfile_discharge = 'discharge.nc'
```

file defining optimazation outputs

Definition at line 67 of file mo_mrm_file.f90.

Referenced by mo_mrm_write::write_daily_obs_sim_discharge().

17.38.2.14 uconfig

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

Unit for file defining mHM's outputs.

Definition at line 60 of file mo_mrm_file.f90.

17.38.2.15 udaily_discharge

```
integer, parameter mo_mrm_file::udaily_discharge = 74
```

Unit for file optimazation outputs.

Definition at line 65 of file mo_mrm_file.f90.

Referenced by mo_mrm_write::write_daily_obs_sim_discharge().

17.38.2.16 udefoutput

```
integer, parameter mo_mrm_file::udefoutput = 67
```

Unit for file defining mRM's outputs.

Definition at line 55 of file mo_mrm_file.f90.

Referenced by mo_mrm_read_config::mrm_read_config().

17.38.2.17 udischarge

```
integer, parameter mo_mrm_file::udischarge = 66
```

unit for discharge time series

Definition at line 50 of file mo_mrm_file.f90.

Referenced by mo_mrm_read_data::mrm_read_discharge().

17.38.2.18 ufacc

```
integer, parameter mo_mrm_file::ufacc = 56
```

Unit for flow accumulation input data file.

Definition at line 34 of file mo_mrm_file.f90.

Referenced by mo_mrm_read_data::mrm_read_l0_data().

17.38.2.19 ufdir

```
integer, parameter mo_mrm_file::ufdir = 57
```

Unit for flow direction input data file.

Definition at line 38 of file mo_mrm_file.f90.

Referenced by mo_mrm_read_data::mrm_read_l0_data().

17.38.2.20 ugaugeloc

```
integer, parameter mo_mrm_file::ugaugeloc = 62
```

Unit for gauge location input data file.

Definition at line 47 of file mo_mrm_file.f90.

Referenced by mo_mrm_read_data::mrm_read_l0_data().

17.38.2.21 unamelist_mrm

```
integer, parameter mo_mrm_file::unamelist_mrm = 40
```

Unit for namelist.

Definition at line 26 of file mo_mrm_file.f90.

17.38.2.22 unamelist_param_mrm

```
integer, parameter mo_mrm_file::unamelist_param_mrm = 41
```

Unit for namelist.

Definition at line 30 of file mo_mrm_file.f90.

17.38.2.23 uslope

```
integer, parameter mo_mrm_file::uslope = 59
```

Unit for flow direction input data file.

Definition at line 42 of file mo_mrm_file.f90.

17.38.2.24 version

```
character(len = *), parameter mo_mrm_file::version = '1.0'
```

Current mHM model version.

Definition at line 18 of file mo_mrm_file.f90.

Referenced by mo_mrm_write_fluxes_states::createoutputfile(), mo_mrm_init::print_startup_message(), and mo_mrm_write::write_configfile().

17.38.2.25 version_date

```
character(len = *), parameter mo_mrm_file::version_date = 'May 2019'
```

Time of current mHM model version release.

Definition at line 20 of file mo_mrm_file.f90.

Referenced by mo_mrm_init::print_startup_message().

17.39 mo_mrm_global_variables Module Reference

Global variables for mRM only.

Data Types

- type gaugingstation
- type domaininfo_mrm

Variables

- logical is_start
- integer(i4) output_deflate_level_mrm
- logical output_double_precision_mrm
- 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 ngaugeslocal
- integer(i4), public ninflowgaugestotal

- integer(i4), public nmeasperday
- type(gaugingstation), public gauge
- type(gaugingstation), public inflowgauge
- type(domaininfo_mrm), dimension(:), allocatable, target, public domain_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
- type(riv_temp_type), public riv_temp_pcs

  This is a container for the river temperature routing process (pcs)

17.39.1 Detailed Description

Global variables for mRM only.

TODO: add description

Authors

Luis Samaniego, Stephan Thober

Date

Aug 2015

17.39.2 Variable Documentation

17.39.2.1 dirbankfullrunoff

```
character(256), dimension(:), allocatable, public mo_mrm_global_variables::dirbankfullrunoff
```

Definition at line 50 of file mo_mrm_global_variables.f90.

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

17.39.2.2 dirgauges

```
character(256), dimension(:), allocatable, public mo_mrm_global_variables::dirgauges
```

Definition at line 46 of file mo_mrm_global_variables.f90.

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().

17.39.2.3 dirtotalrunoff

```
character(256), dimension(:), allocatable, public mo_mrm_global_variables::dirtotalrunoff
```

Definition at line 47 of file mo_mrm_global_variables.f90.

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

17.39.2.4 domain_mrm

```
type(domaininfo_mrm), dimension(:), allocatable, target, public mo_mrm_global_variables::domain_mrm
```

Definition at line 116 of file mo_mrm_global_variables.f90.

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_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().

17.39.2.5 filenametotalrunoff

```
character(256), public mo_mrm_global_variables::filenametotalrunoff
```

Definition at line 48 of file mo_mrm_global_variables.f90.

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

17.39.2.6 gauge

```
type(gaugingstation), public mo_mrm_global_variables::gauge
```

Definition at line 87 of file mo_mrm_global_variables.f90.

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().

17.39.2.7 gw_coupling

```
logical mo_mrm_global_variables::gw_coupling
```

Definition at line 215 of file mo_mrm_global_variables.f90.

Referenced by mo_mhm_eval::mhm_eval(), mo_mrm_init::mrm_init(), and mo_mrm_read_config::mrm_read_↩
config().

17.39.2.8 inflowgauge

```
type(gaugingstation), public mo_mrm_global_variables::inflowgauge
```

Definition at line 88 of file mo_mrm_global_variables.f90.

Referenced by mo_mhm_eval::mhm_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().

17.39.2.9 is_start

```
logical mo_mrm_global_variables::is_start
```

Definition at line 30 of file mo_mrm_global_variables.f90.

Referenced by mo_mrm_read_config::mrm_read_config(), and mo_mrm_routing::mrm_routing().

17.39.2.10 l0_celerity

```
real(dp), dimension(:), allocatable, public mo_mrm_global_variables::l0_celerity
```

Definition at line 136 of file mo_mrm_global_variables.f90.

Referenced by mo_mrm_init::variables_alloc_routing().

17.39.2.11 l0_channel_depth

```
real(dp), dimension(:), allocatable, public mo_mrm_global_variables::l0_channel_depth
```

Definition at line 219 of file mo_mrm_global_variables.f90.

Referenced by mo_mrm_river_head::calc_channel_elevation().

17.39.2.12 l0_channel_elevation

```
real(dp), dimension(:), allocatable, public mo_mrm_global_variables::l0_channel_elevation
```

Definition at line 220 of file mo_mrm_global_variables.f90.

Referenced by mo_mrm_river_head::calc_river_head().

17.39.2.13 l0_dracell

```
integer(i4), dimension(:), allocatable, public mo_mrm_global_variables::l0_dracell
```

Definition at line 132 of file mo_mrm_global_variables.f90.

Referenced by mo_mrm_net_startup::l11_set_drain_outlet_gauges().

17.39.2.14 l0_drasc

```
integer(i4), dimension(:), allocatable, public mo_mrm_global_variables::l0_drasc
```

Definition at line 130 of file mo_mrm_global_variables.f90.

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().

17.39.2.15 l0_facc

```
integer(i4), dimension(:), allocatable, public mo_mrm_global_variables::l0_facc
```

Definition at line 125 of file mo_mrm_global_variables.f90.

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().

17.39.2.16 l0_fdir

```
integer(i4), dimension(:), allocatable, public mo_mrm_global_variables::l0_fdir
```

Definition at line 126 of file mo_mrm_global_variables.f90.

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().

17.39.2.17 l0_floodplain

```
integer(i4), dimension(:), allocatable, public mo_mrm_global_variables::l0_floodplain
```

Definition at line 134 of file mo_mrm_global_variables.f90.

Referenced by mo_mrm_net_startup::l11_fraction_sealed_floodplain(), and mo_mrm_net_startup::l11_stream_↩features().

17.39.2.18 l0_gaugeloc

```
integer(i4), dimension(:), allocatable, public mo_mrm_global_variables::l0_gaugeloc
```

Definition at line 123 of file mo_mrm_global_variables.f90.

Referenced by mo_mrm_net_startup::l11_set_drain_outlet_gauges(), and mo_mrm_read_data::mrm_read_l0_↩data().

17.39.2.19 l0_inflowgaugeloc

```
integer(i4), dimension(:), allocatable, public mo_mrm_global_variables::l0_inflowgaugeloc
```

Definition at line 124 of file mo_mrm_global_variables.f90.

Referenced by mo_mrm_net_startup::l11_set_drain_outlet_gauges(), and mo_mrm_read_data::mrm_read_l0_↩data().

17.39.2.20 l0_l11_remap

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

Definition at line 62 of file mo_mrm_global_variables.f90.

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().

17.39.2.21 l0_noutlet

```
integer(i4), dimension(:), allocatable, public mo_mrm_global_variables::l0_noutlet
```

Definition at line 135 of file mo_mrm_global_variables.f90.

17.39.2.22 l0_river_head_mon_sum

```
real(dp), dimension(:), allocatable, public mo_mrm_global_variables::l0_river_head_mon_sum
```

Definition at line 222 of file mo_mrm_global_variables.f90.

Referenced by mo_mhm_eval::mhm_eval(), and mo_mrm_init::mrm_init().

17.39.2.23 l0_slope

```
real(dp), dimension(:), allocatable, public mo_mrm_global_variables::l0_slope
```

Definition at line 223 of file mo_mrm_global_variables.f90.

Referenced by mo_mrm_river_head::calc_river_head().

17.39.2.24 l0_streamnet

```
integer(i4), dimension(:), allocatable, public mo_mrm_global_variables::l0_streamnet
```

Definition at line 133 of file mo_mrm_global_variables.f90.

Referenced by mo_mrm_net_startup::l11_stream_features(), mo_mrm_restart::mrm_read_restart_config(), and mo_mrm_restart::mrm_write_restart().

17.39.2.25 l11_afloodplain

```
real(dp), dimension(:), allocatable, target, public mo_mrm_global_variables::l11_afloodplain
```

Definition at line 195 of file mo_mrm_global_variables.f90.

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().

17.39.2.26 l11_areacell

```
real(dp), dimension(:), allocatable, public mo_mrm_global_variables::l11_areacell
```

Definition at line 158 of file mo_mrm_global_variables.f90.

17.39.2.27 l11_bankfull_runoff_in

```
real(dp), dimension(:), allocatable, public mo_mrm_global_variables::l11_bankfull_runoff_in
```

Definition at line 217 of file mo_mrm_global_variables.f90.

Referenced by mo_mrm_river_head::calc_river_head().

17.39.2.28 l11_c1

```
real(dp), dimension(:), allocatable, public mo_mrm_global_variables::l11_c1
```

Definition at line 208 of file mo_mrm_global_variables.f90.

Referenced by mo_mhm_eval::mhm_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().

17.39.2.29 l11_c2

```
real(dp), dimension(:), allocatable, public mo_mrm_global_variables::l11_c2
```

Definition at line 209 of file mo_mrm_global_variables.f90.

Referenced by mo_mhm_eval::mhm_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().

17.39.2.30 l11_celerity

```
real(dp), dimension(:), allocatable, public mo_mrm_global_variables::l11_celerity
```

Definition at line 162 of file mo_mrm_global_variables.f90.

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().

17.39.2.31 l11_cellcoor

```
integer(i4), dimension(:,:), allocatable, public mo_mrm_global_variables::l11_cellcoor
```

Definition at line 155 of file mo_mrm_global_variables.f90.

17.39.2.32 l11_colout

```
integer(i4), dimension(:), allocatable, public mo_mrm_global_variables::l11_colout
```

Definition at line 171 of file mo_mrm_global_variables.f90.

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().

17.39.2.33 l11_facc

```
real(dp), dimension(:), allocatable, public mo_mrm_global_variables::l11_facc
```

Definition at line 159 of file mo_mrm_global_variables.f90.

Referenced by mo_mrm_restart::mrm_read_restart_config(), and mo_mrm_restart::mrm_write_restart().

17.39.2.34 l11_fcol

```
integer(i4), dimension(:), allocatable, public mo_mrm_global_variables::l11_fcol
```

Definition at line 188 of file mo_mrm_global_variables.f90.

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().

17.39.2.35 l11_fdir

```
integer(i4), dimension(:), allocatable, public mo_mrm_global_variables::l11_fdir
```

Definition at line 160 of file mo_mrm_global_variables.f90.

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().

17.39.2.36 l11_fromn

```
integer(i4), dimension(:), allocatable, public mo_mrm_global_variables::l11_fromn
```

Definition at line 184 of file mo_mrm_global_variables.f90.

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_init::mrm_init(), mo_mrm_↩restart::mrm_read_restart_config(), mo_mrm_restart::mrm_write_restart(), mo_write_ascii::write_configfile(), and mo_mrm_write::write_configfile().

17.39.2.37 l11_frow

```
integer(i4), dimension(:), allocatable, public mo_mrm_global_variables::l11_frow
```

Definition at line 187 of file mo_mrm_global_variables.f90.

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().

17.39.2.38 l11_k

```
real(dp), dimension(:), allocatable, public mo_mrm_global_variables::l11_k
```

Definition at line 204 of file mo_mrm_global_variables.f90.

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().

17.39.2.39 l11_l1_id

```
integer(i4), dimension(:), allocatable, public mo_mrm_global_variables::l11_l1_id
```

Definition at line 142 of file mo_mrm_global_variables.f90.

Referenced by mo_mrm_net_startup::l11_l1_mapping(), mo_mhm_eval::mhm_eval(), mo_mrm_restart::mrm_↩ read_restart_config(), mo_mrm_restart::mrm_write_restart(), and mo_mrm_write::write_configfile().

17.39.2.40 l11_label

```
integer(i4), dimension(:), allocatable, public mo_mrm_global_variables::l11_label
```

Definition at line 192 of file mo_mrm_global_variables.f90.

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().

17.39.2.41 l11_length

```
real(dp), dimension(:), allocatable, public mo_mrm_global_variables::l11_length
```

Definition at line 194 of file mo_mrm_global_variables.f90.

Referenced by mo_mrm_net_startup::l11_stream_features(), mo_mhm_eval::mhm_eval(), mo_mrm_init::mrm↩ _init(), 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().

17.39.2.42 l11_linkin_facc

```
real(dp), dimension(:), allocatable, public mo_mrm_global_variables::l11_linkin_facc
```

Definition at line 166 of file mo_mrm_global_variables.f90.

17.39.2.43 l11_meandering

```
real(dp), dimension(:), allocatable, public mo_mrm_global_variables::l11_meandering
```

Definition at line 164 of file mo_mrm_global_variables.f90.

17.39.2.44 l11_netperm

```
integer(i4), dimension(:), allocatable, public mo_mrm_global_variables::l11_netperm
```

Definition at line 186 of file mo_mrm_global_variables.f90.

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_init::mrm_init(), mo_mrm_restart↩ ::mrm_read_restart_config(), mo_mrm_restart::mrm_write_restart(), mo_write_ascii::write_configfile(), and mo_↩ mrm_write::write_configfile().

17.39.2.45 l11_nlinkfracfpimp

```
real(dp), dimension(:, :), allocatable, public mo_mrm_global_variables::l11_nlinkfracfpimp
```

Definition at line 201 of file mo_mrm_global_variables.f90.

Referenced by mo_mrm_net_startup::l11_fraction_sealed_floodplain(), mo_mhm_eval::mhm_eval(), mo_mrm_↩ restart::mrm_read_restart_states(), and mo_mrm_restart::mrm_write_restart().

17.39.2.46 l11_noutlets

```
integer(i4), dimension(:), allocatable, public mo_mrm_global_variables::l11_noutlets
```

Definition at line 161 of file mo_mrm_global_variables.f90.

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_init::mrm_init(), mo_mrm_restart::mrm_read_restart_↩ config(), mo_mrm_mpr::mrm_update_param(), and mo_write_ascii::write_configfile().

17.39.2.47 l11_qmod

```
real(dp), dimension(:), allocatable, public mo_mrm_global_variables::l11_qmod
```

Definition at line 179 of file mo_mrm_global_variables.f90.

Referenced by mo_mhm_eval::mhm_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().

17.39.2.48 l11_qout

```
real(dp), dimension(:), allocatable, public mo_mrm_global_variables::l11_qout
```

Definition at line 180 of file mo_mrm_global_variables.f90.

Referenced by mo_mhm_eval::mhm_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().

17.39.2.49 l11_qtin

```
real(dp), dimension(:, :), allocatable, public mo_mrm_global_variables::l11_qtin
```

Definition at line 181 of file mo_mrm_global_variables.f90.

Referenced by mo_mhm_eval::mhm_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().

17.39.2.50 l11_qtr

```
real(dp), dimension(:, :), allocatable, public mo_mrm_global_variables::l11_qtr
```

Definition at line 182 of file mo_mrm_global_variables.f90.

Referenced by mo_mhm_eval::mhm_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().

17.39.2.51 l11_rorder

```
integer(i4), dimension(:), allocatable, public mo_mrm_global_variables::l11_rorder
```

Definition at line 191 of file mo_mrm_global_variables.f90.

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().

17.39.2.52 l11_rowout

```
integer(i4), dimension(:), allocatable, public mo_mrm_global_variables::l11_rowout
```

Definition at line 170 of file mo_mrm_global_variables.f90.

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().

17.39.2.53 l11_sink

```
logical, dimension(:), allocatable, public mo_mrm_global_variables::l11_sink
```

Definition at line 193 of file mo_mrm_global_variables.f90.

Referenced by mo_mrm_net_startup::l11_routing_order(), mo_mrm_restart::mrm_read_restart_config(), and mo↩ _mrm_restart::mrm_write_restart().

17.39.2.54 l11_slope

```
real(dp), dimension(:), allocatable, public mo_mrm_global_variables::l11_slope
```

Definition at line 197 of file mo_mrm_global_variables.f90.

Referenced by mo_mrm_net_startup::l11_stream_features(), mo_mhm_eval::mhm_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().

17.39.2.55 l11_tcol

```
integer(i4), dimension(:), allocatable, public mo_mrm_global_variables::l11_tcol
```

Definition at line 190 of file mo_mrm_global_variables.f90.

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().

17.39.2.56 l11_ton

```
integer(i4), dimension(:), allocatable, public mo_mrm_global_variables::l11_ton
```

Definition at line 185 of file mo_mrm_global_variables.f90.

Referenced by mo_mrm_net_startup::l11_routing_order(), mo_mrm_net_startup::l11_set_network_topology(), mo_mhm_eval::mhm_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().

17.39.2.57 l11_trow

```
integer(i4), dimension(:), allocatable, public mo_mrm_global_variables::l11_trow
```

Definition at line 189 of file mo_mrm_global_variables.f90.

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().

17.39.2.58 l11_tsrout

```
real(dp), dimension(:), allocatable, public mo_mrm_global_variables::l11_tsrout
```

Definition at line 207 of file mo_mrm_global_variables.f90.

Referenced by mo_mhm_eval::mhm_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().

17.39.2.59 l11_xi

```
real(dp), dimension(:), allocatable, public mo_mrm_global_variables::l11_xi
```

Definition at line 205 of file mo_mrm_global_variables.f90.

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().

17.39.2.60 l1_l11_id

```
integer(i4), dimension(:), allocatable, public mo_mrm_global_variables::l1_l11_id
```

Definition at line 157 of file mo_mrm_global_variables.f90.

Referenced by mo_mrm_net_startup::l11_l1_mapping(), mo_mhm_eval::mhm_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().

17.39.2.61 l1_l11_remap

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

Definition at line 63 of file mo_mrm_global_variables.f90.

Referenced by mo_mrm_init::mrm_init().

17.39.2.62 l1_total_runoff_in

```
real(dp), dimension(:, :), allocatable, public mo_mrm_global_variables::l1_total_runoff_in
```

Definition at line 148 of file mo_mrm_global_variables.f90.

Referenced by mo_mrm_read_data::mrm_read_total_runoff().

17.39.2.63 level11

```
type(grid), dimension(:), allocatable, target, public mo_mrm_global_variables::level11
```

Definition at line 61 of file mo_mrm_global_variables.f90.

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_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().

17.39.2.64 mrm_runoff

```
real(dp), dimension(:, :), allocatable, public mo_mrm_global_variables::mrm_runoff
```

Definition at line 69 of file mo_mrm_global_variables.f90.

Referenced by mo_mhm_eval::mhm_eval(), mo_mrm_read_data::mrm_read_discharge(), and mo_mrm_write::mrm_write().

17.39.2.65 ngaugeslocal

```
integer(i4), public mo_mrm_global_variables::ngaugeslocal
```

Definition at line 75 of file mo_mrm_global_variables.f90.

Referenced by mo_mrm_read_config::mrm_read_config(), mo_mrm_read_data::mrm_read_discharge(), and mo_write_ascii::write_configfile().

17.39.2.66 ngaugestotal

```
integer(i4), public mo_mrm_global_variables::ngaugestotal
```

Definition at line 74 of file mo_mrm_global_variables.f90.

Referenced by mo_mrm_read_config::mrm_read_config(), mo_mrm_write::mrm_write(), mo_mrm_objective_function_runoff::single_objective_runoff_master(), mo_write_ascii::write_configfile(), and mo_mrm_write::write_configfile().

17.39.2.67 ninflowgaugestotal

```
integer(i4), public mo_mrm_global_variables::ninflowgaugestotal
```

Definition at line 76 of file mo_mrm_global_variables.f90.

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().

17.39.2.68 nmeasperday

```
integer(i4), public mo_mrm_global_variables::nmeasperday
```

Definition at line 77 of file mo_mrm_global_variables.f90.

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().

17.39.2.69 ntstepday

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

Definition at line 55 of file mo_mrm_global_variables.f90.

17.39.2.70 output_deflate_level_mrm

```
integer(i4) mo_mrm_global_variables::output_deflate_level_mrm
```

Definition at line 36 of file mo_mrm_global_variables.f90.

Referenced by mo_mrm_read_config::mrm_read_config(), and mo_mrm_write_fluxes_states::newoutputvariable().

17.39.2.71 output_double_precision_mrm

```
logical mo_mrm_global_variables::output_double_precision_mrm
```

Definition at line 37 of file mo_mrm_global_variables.f90.

Referenced by mo_mrm_read_config::mrm_read_config().

17.39.2.72 outputflxstate_mrm

```
logical, dimension(noutflxstate) mo_mrm_global_variables::outputflxstate_mrm
```

Definition at line 39 of file mo_mrm_global_variables.f90.

Referenced by mo_mhm_eval::mhm_eval(), mo_mrm_read_config::mrm_read_config(), mo_mrm_write_fluxes_↩ states::newoutputdataset(), and mo_mrm_write_fluxes_states::updatedataset().

17.39.2.73 riv_temp_pcs

```
type(riv_temp_type), public mo_mrm_global_variables::riv_temp_pcs
```

This is a container for the river temperature routing process (pcs)

Definition at line 229 of file mo_mrm_global_variables.f90.

Referenced by mo_mhm_eval::mhm_eval(), mo_mrm_init::mrm_configuration(), mo_mrm_init::mrm_init(), mo_mrm_read_data::mrm_read_discharge(), mo_mrm_routing::mrm_routing(), mo_mrm_write::mrm_write_output_fluxes(), and mo_mrm_write_fluxes_states::newoutputdataset().

17.39.2.74 timestep_model_outputs_mrm

```
integer(i4) mo_mrm_global_variables::timestep_model_outputs_mrm
```

Definition at line 38 of file mo_mrm_global_variables.f90.

Referenced by mo_mhm_eval::mhm_eval(), and mo_mrm_read_config::mrm_read_config().

17.39.2.75 varnametotalrunoff

```
character(256), public mo_mrm_global_variables::varnametotalrunoff
```

Definition at line 49 of file mo_mrm_global_variables.f90.

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

17.40 mo_mrm_init Module Reference

Wrapper for initializing Routing.

Functions/Subroutines

- subroutine, public mrm_configuration (file_namelist, unamelist, file_namelist_param, unamelist_param, ReadLatLon)
- subroutine, public mrm_init (file_namelist, unamelist, file_namelist_param, unamelist_param, ReadLatLon)

  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 (L0Domain_iDomain)

  TODO: add description.
- subroutine variables_alloc_routing (iDomain)

  TODO: add description.

17.40.1 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

17.40.2 Function/Subroutine Documentation

17.40.2.1 config_output()

```
subroutine mo_mrm_init::config_output
```

TODO: add description.

TODO: add description

Authors

Robert Schweppe

Date

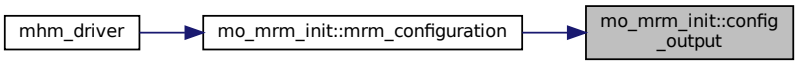
Jun 2018

Definition at line 412 of file mo_mrm_init.f90.

References mo_mrm_global_variables::dirgauges, mo_common_variables::dirlcover, mo_common_variables↩ ::dirmorpho, mo_common_variables::dirout, mo_mrm_global_variables::domain_mrm, mo_common_variables↩ ::domainmeta, mo_mrm_file::file_defoutput, mo_mrm_file::file_namelist_mrm, and mo_mrm_file::file_namelist_↩ param_mrm.

Referenced by mrm_configuration().

Here is the caller graph for this function:



17.40.2.2 l0_check_input_routing()

```
subroutine mo_mrm_init::l0_check_input_routing (
            integer(i4), intent(in) L0Domain_iDomain )
```

TODO: add description.

TODO: add description

Parameters

in	*integer(i4) :: L0Domain_iDomain*	

Authors

Robert Schweppe

Date

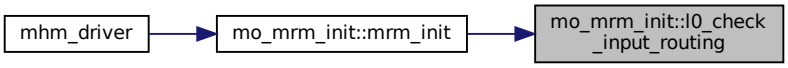
Jun 2018

Definition at line 548 of file mo_mrm_init.f90.

References mo_mrm_global_variables::l0_facc, mo_mrm_global_variables::l0_fdir, mo_common_variables::level0, and mo_common_constants::nodata_i4.

Referenced by mrm_init().

Here is the caller graph for this function:



17.40.2.3 mrm_configuration()

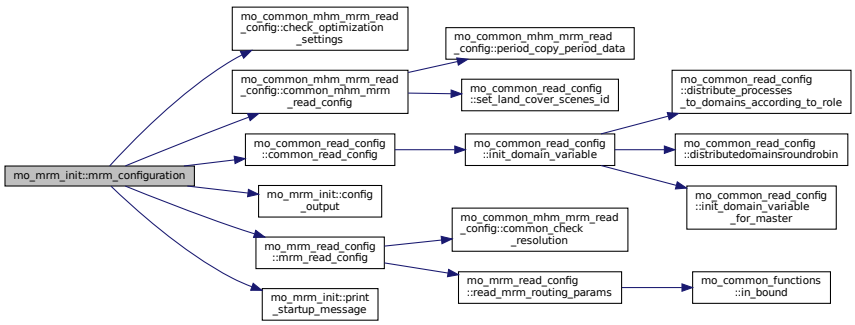
```
subroutine, public mo_mrm_init::mrm_configuration (
             character(*), intent(in) file_namelist,
             integer, intent(in) unamelist,
             character(*), intent(in) file_namelist_param,
             integer, intent(in) unamelist_param,
             logical, intent(out) ReadLatLon )
```

Definition at line 30 of file mo_mrm_init.f90.

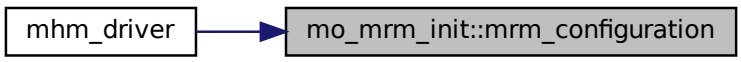
References mo_common_mhm_mrm_read_config::check_optimization_settings(), mo_common_mhm_mrm_↩read_config::common_mhm_mrm_read_config(), mo_common_read_config::common_read_config(), config_↩output(), mo_common_mhm_mrm_variables::mrm_coupling_mode, mo_mrm_read_config::mrm_read_config(), print_startup_message(), mo_common_variables::processmatrix, and mo_mrm_global_variables::riv_temp_pcs.

Referenced by mhm_driver().

Here is the call graph for this function:



Here is the caller graph for this function:



17.40.2.4 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,
             logical, intent(inout), optional ReadLatLon )
```

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

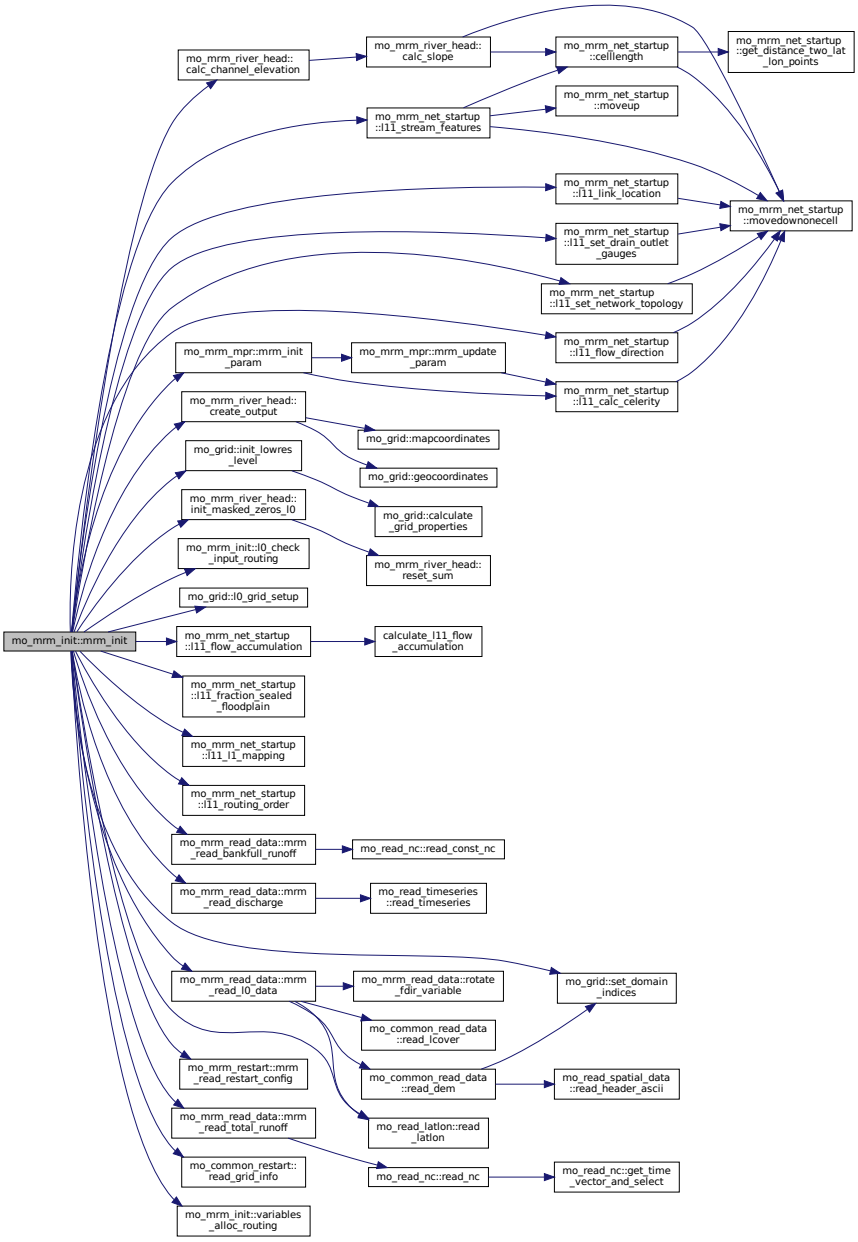
Aug 2015

Definition at line 113 of file mo_mrm_init.f90.

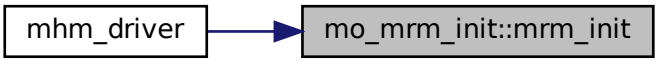
References mo_mrm_river_head::calc_channel_elevation(), mo_mrm_river_head::create_output(), mo_mrm_global_variables::domain_mrm, mo_common_variables::domainmeta, mo_common_variables::global_parameters, mo_mrm_global_variables::gw_coupling, mo_grid::init_lowres_level(), mo_mrm_river_head::init_masked_zeros_l0(), 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_global_variables::l11_fromn, mo_mrm_net_startup::l11_l1_mapping(), mo_mrm_global_variables::l11_length, mo_mrm_net_startup::l11_link_location(), mo_mrm_global_variables::l11_netperm, mo_mrm_global_variables::l11_noutlets, 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_common_mhm_mrm_variables::mrm_coupling_mode, mo_mrm_mpr::mrm_init_param(), mo_mrm_read_data::mrm_read_bankfull_runoff(), mo_mrm_read_data::mrm_read_discharge(), mo_mrm_read_data::mrm_read_l0_data(), mo_mrm_restart::mrm_read_restart_config(), mo_common_mhm_mrm_variables::mrm_read_river_network, mo_mrm_read_data::mrm_read_total_runoff(), mo_common_mhm_mrm_variables::mrmfilerestartin, mo_common_constants::nodata_dp, mo_common_constants::nodata_i4, mo_common_variables::processmatrix, mo_common_restart::read_grid_info(), mo_read_latlon::read_latlon(), mo_common_variables::resolutionhydrology, mo_common_mhm_mrm_variables::resolutionrouting, mo_mrm_global_variables::riv_temp_pcs, mo_grid::set_domain_indices(), and variables_alloc_routing().

Referenced by mhm_driver().

Here is the call graph for this function:



Here is the caller graph for this function:



17.40.2.5 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

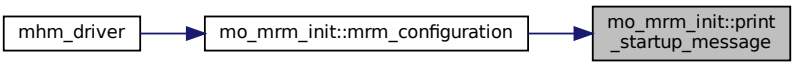
Jun 2018

Definition at line 351 of file mo_mrm_init.f90.

References mo_mrm_file::file_defoutput, mo_mrm_file::file_main, mo_mrm_file::version, and mo_mrm_file↩ ::version_date.

Referenced by mrm_configuration().

Here is the caller graph for this function:



17.40.2.6 variables_alloc_routing()

```
subroutine mo_mrm_init::variables_alloc_routing (
            integer(i4), intent(in) iDomain )
```

TODO: add description.

TODO: add description

Parameters

in	*integer(i4) :: iDomain*	

Authors

Robert Schweppe

Date

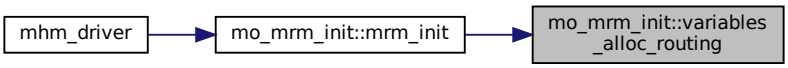
Jun 2018

Definition at line 606 of file mo_mrm_init.f90.

References mo_common_variables::domainmeta, 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:



17.40.2.7 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

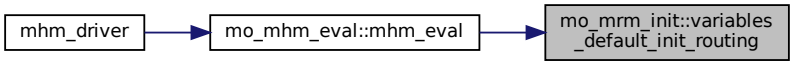
Jun 2018

Definition at line 495 of file mo_mrm_init.f90.

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().

Here is the caller graph for this function:



17.41 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 (iDomain, param)

  TODO: add description.
- subroutine, public mrm_update_param (iDomain, param)

  TODO: add description.

17.41.1 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

17.41.2 Function/Subroutine Documentation

17.41.2.1 mrm_init_param()

```
subroutine, public mo_mrm_mpr::mrm_init_param (
            integer(i4), intent(in) iDomain,
            real(dp), dimension(:), intent(in) param )
```

TODO: add description.

TODO: add description

Parameters

in	*integer(i4) :: iDomain*	domain number
in	*real(dp), dimension(:) :: param*	input parameter (param(1) is celerity in m/s)

Authors

Stephan Thober, Matthias Kelbling

Date

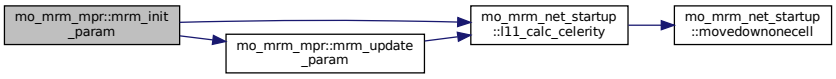
Jun 2018

Definition at line 144 of file mo_mrm_mpr.f90.

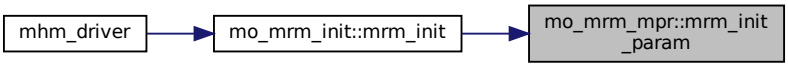
References mo_mrm_global_variables::domain_mrm, mo_common_variables::domainmeta, mo_mrm_constants::given_ts, mo_common_variables::iflag_cordinate_sys, mo_mrm_net_startup::l11_calc_celerity(), mo_mrm_global_variables::l11_celerity, mo_mrm_global_variables::l11_tsrout, mo_mrm_global_variables::level11, mrm_update_param(), 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:



17.41.2.2 mrm_update_param()

```
subroutine, public mo_mrm_mpr::mrm_update_param (
            integer(i4), intent(in) iDomain,
            real(dp), dimension(1), intent(in) param )
```

TODO: add description.

TODO: add description

Parameters

in	*integer(i4) :: iDomain*	domain number
in	*real(dp), dimension(1) :: param*	celerity parameter [m s-1]

Authors

Stehpan Thober, Matthias Kelbling

Date

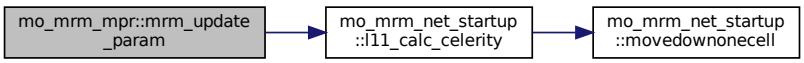
Jun 2018

Definition at line 241 of file mo_mrm_mpr.f90.

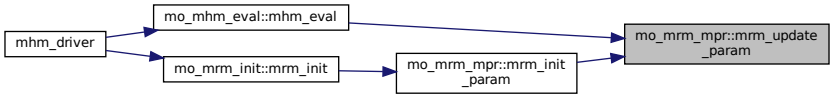
References mo_mrm_constants::given_ts, mo_common_variables::iflag_cordinate_sys, mo_mrm_global_↩ variables::l11_c1, mo_mrm_global_variables::l11_c2, mo_mrm_net_startup::l11_calc_celerity(), mo_mrm_↩ global_variables::l11_celerity, mo_mrm_global_variables::l11_length, mo_mrm_global_variables::l11_noutlets, mo_mrm_global_variables::l11_tsrout, mo_mrm_global_variables::level11, mo_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(), and mrm_init_param().

Here is the call graph for this function:



Here is the caller graph for this function:



17.41.2.3 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

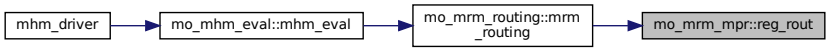
Date

Dec 2012

Definition at line 60 of file mo_mrm_mpr.f90.

Referenced by mo_mrm_routing::mrm_routing().

Here is the caller graph for this function:



17.42 mo_mrm_net_startup Module Reference

Startup drainage network for mHM.

Functions/Subroutines

- subroutine, public l11_l1_mapping (iDomain)

  TODO: add description.
- subroutine, public l11_flow_direction (iDomain)

Determine the flow direction of the upscaled river network at level L11.

- subroutine, public l11_set_network_topology (iDomain)

  Set network topology.
- subroutine, public l11_routing_order (iDomain)

  Find routing order, headwater cells and sink.
- subroutine, public l11_link_location (iDomain)

  Estimate the LO (row,col) location for each routing link at level L11.
- subroutine, public l11_set_drain_outlet_gauges (iDomain)

  Draining cell identification and Set gauging node.
- subroutine, public l11_stream_features (iDomain)

  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 movedownnonecell (fDir, iRow, jCol)

  TODO: add description.
- subroutine celllength (iDomain, 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 (iDomain)

  Calculates L11 flow accumulation per grid cell.
- subroutine, public l11_calc_celerity (iDomain, param)

  L11 celerity based on L0 elevation and L0 fAcc.

17.42.1 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

17.42.2 Function/Subroutine Documentation

17.42.2.1 celllength()

```
subroutine mo_mrm_net_startup::celllength (
             integer(i4), intent(in) iDomain,
             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) :: iDomain*	
in	*integer(i4) :: fDir*	
in	*integer(i4) :: iRow*	
in	*integer(i4) :: jCol*	
in	*integer(i4) :: iCoorSystem*	
out	*real(dp) :: length*	

Authors

Robert Schweppe

Date

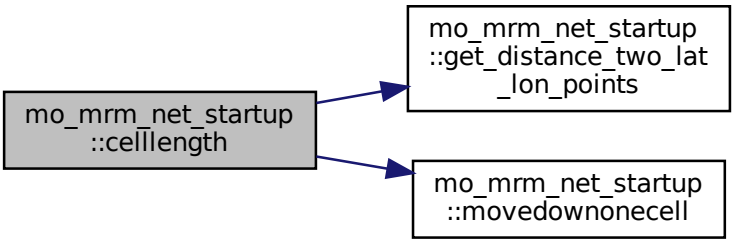
Jun 2018

Definition at line 1823 of file mo_mrm_net_startup.f90.

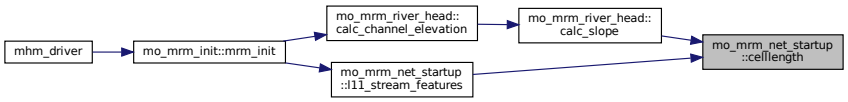
References mo_common_variables::domainmeta, get_distance_two_lat_lon_points(), mo_common_variables↩ ::level0, and movedownonecell().

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:



17.42.2.2 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

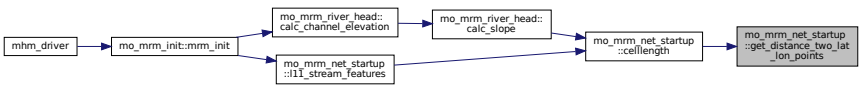
Date

May 2014

Definition at line 1917 of file mo_mrm_net_startup.f90.

Referenced by celllength().

Here is the caller graph for this function:



17.42.2.3 l11_calc_celerity()

```
subroutine, public mo_mrm_net_startup::l11_calc_celerity (
            integer(i4), intent(in) iDomain,
            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

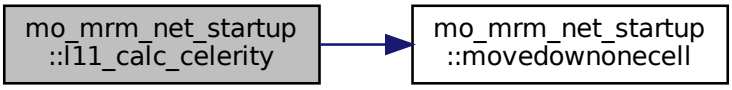
in		

Definition at line 2203 of file mo_mrm_net_startup.f90.

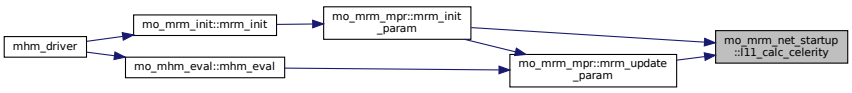
References mo_common_variables::domainmeta, mo_mrm_global_variables::l0_fdir, mo_mpr_global_variables::l0_slope, movedownonecell(), mo_common_constants::nodata_dp, and mo_common_constants::nodata_i4.

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:



17.42.2.4 l11_flow_accumulation()

```
subroutine, public mo_mrm_net_startup::l11_flow_accumulation (
            integer(i4), intent(in) iDomain )
```

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 iDomain

Author

Matthias Kelbling

Date

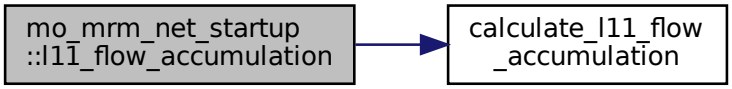
Aug 2017

Definition at line 2013 of file mo_mrm_net_startup.f90.

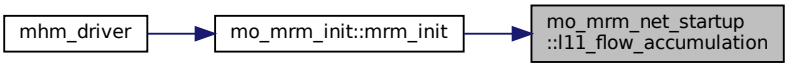
References calculate_l11_flow_accumulation(), mo_mrm_global_variables::l11_fdir, mo_mrm_global_variables::level11, mo_common_constants::nodata_dp, and mo_common_constants::nodata_i4.

Referenced by mo_mrm_init::mrm_init().

Here is the call graph for this function:



Here is the caller graph for this function:



17.42.2.5 l11_flow_direction()

```
subroutine, public mo_mrm_net_startup::l11_flow_direction (
             integer(i4), intent(in) iDomain )
```

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.

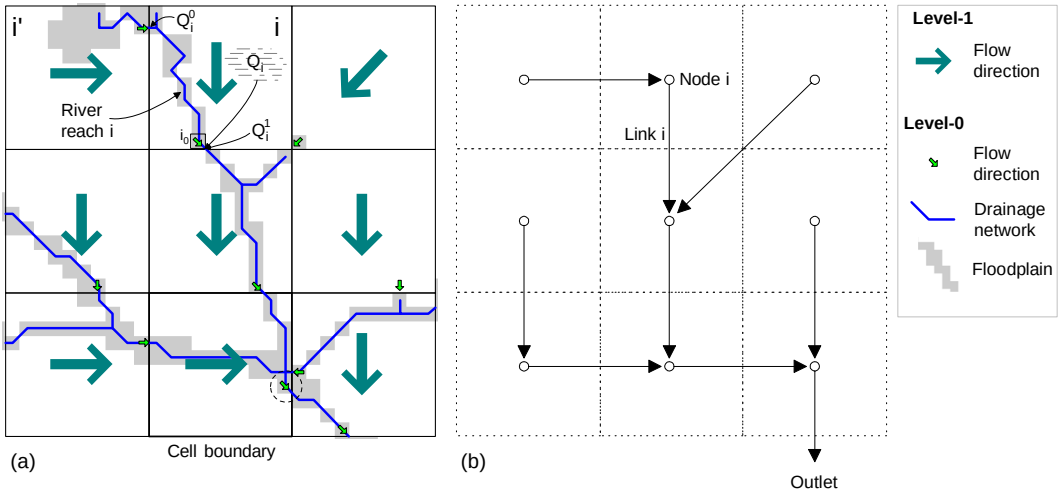


Figure 17.1: Upscaling routing network from L0 to L1 (or L11)

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) :: iDomain*	Domain Id

Authors

Luis Samaniego

Date

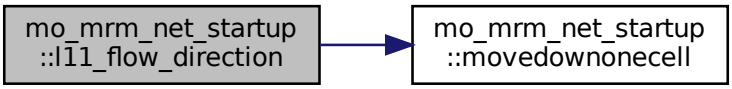
Dec 2005

Definition at line 221 of file mo_mrm_net_startup.f90.

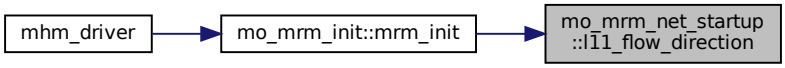
References mo_mrm_global_variables::domain_mrm, mo_common_variables::domainmeta, mo_mrm_global_variables::l0_drasc, mo_mrm_global_variables::l0_facc, mo_mrm_global_variables::l0_fdir, mo_mrm_global_variables::l0_l11_remap, mo_mrm_global_variables::l11_colout, mo_mrm_global_variables::l11_fdir, mo_mrm_global_variables::l11_noutlets, mo_mrm_global_variables::l11_rowout, mo_common_variables::level0, mo_mrm_global_variables::level11, 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:



17.42.2.6 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 Domains. 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

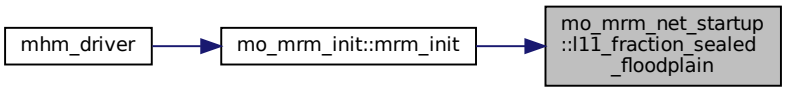
Definition at line 1503 of file mo_mrm_net_startup.f90.

References mo_common_variables::domainmeta, 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::nlcoverscene, and mo_common_constants::nodata_dp.

Referenced by mo_mrm_init::mrm_init().

Here is the caller graph for this function:



17.42.2.7 l11_l1_mapping()

```
subroutine, public mo_mrm_net_startup::l11_l1_mapping (
             integer(i4), intent(in) iDomain )
```

TODO: add description.

TODO: add description

Parameters

in	*integer(i4) :: iDomain*	domain

Authors

Robert Schweppe

Date

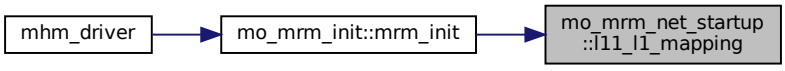
Jun 2018

Definition at line 55 of file mo_mrm_net_startup.f90.

References 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:



17.42.2.8 l11_link_location()

```
subroutine, public mo_mrm_net_startup::l11_link_location (
             integer(i4), intent(in) iDomain )
```

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) :: iDomain*	Domain Id

Authors

Luis Samaniego

Date

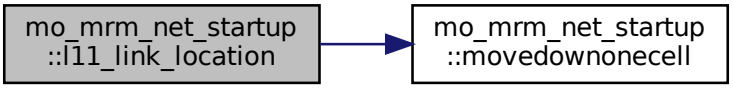
Dec 2005

Definition at line 881 of file mo_mrm_net_startup.f90.

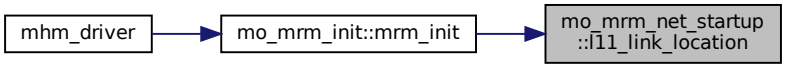
References mo_mrm_global_variables::domain_mrm, mo_common_variables::domainmeta, mo_mrm_global_↩ variables::l0_drasc, mo_mrm_global_variables::l0_fdir, mo_mrm_global_variables::l11_colout, mo_mrm_global_↩ variables::l11_fcol, mo_mrm_global_variables::l11_fromn, mo_mrm_global_variables::l11_frow, mo_mrm_global↩ _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, 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:



17.42.2.9 l11_routing_order()

```
subroutine, public mo_mrm_net_startup::l11_routing_order (
            integer(i4), intent(in) iDomain )
```

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) :: iDomain*	Domain Id

Authors

Luis Samaniego

Date

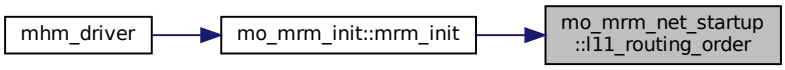
Dec 2005

Definition at line 722 of file mo_mrm_net_startup.f90.

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:



17.42.2.10 l11_set_drain_outlet_gauges()

```
subroutine, public mo_mrm_net_startup::l11_set_drain_outlet_gauges (
            integer(i4), intent(in) iDomain )
```

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) :: iDomain*	Domain Id

Authors

Luis Samaniego

Date

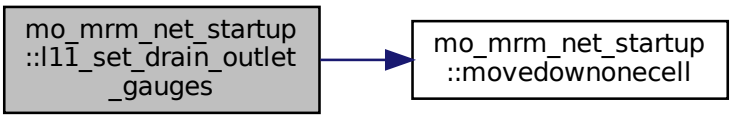
Dec 2005

Definition at line 1083 of file mo_mrm_net_startup.f90.

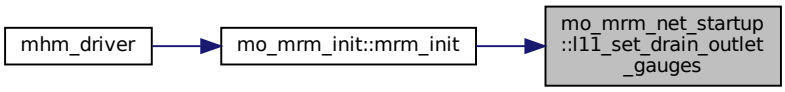
References mo_mrm_global_variables::domain_mrm, mo_common_variables::domainmeta, mo_mrm_global_↩ variables::l0_dracell, mo_mrm_global_variables::l0_drasc, mo_mrm_global_variables::l0_fdir, mo_mrm_global↩ _variables::l0_gaugeloc, mo_mrm_global_variables::l0_inflowgaugeloc, mo_mrm_global_variables::l0_l11_remap, 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:



17.42.2.11 l11_set_network_topology()

```
subroutine, public mo_mrm_net_startup::l11_set_network_topology (
            integer(i4), intent(in) iDomain )
```

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) :: iDomain*	Domain Id

Authors

Luis Samaniego

Date

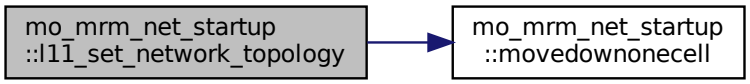
Dec 2005

Definition at line 624 of file mo_mrm_net_startup.f90.

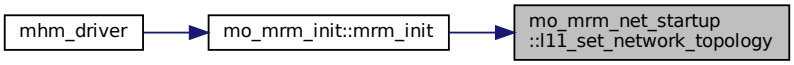
References mo_mrm_global_variables::l11_fdir, mo_mrm_global_variables::l11_fromn, mo_mrm_global_variables::l11_ton, mo_mrm_global_variables::level11, movedownonecell(), and mo_common_constants::nodata_i4.

Referenced by mo_mrm_init::mrm_init().

Here is the call graph for this function:



Here is the caller graph for this function:



17.42.2.12 l11_stream_features()

```
subroutine, public mo_mrm_net_startup::l11_stream_features (
            integer(i4), intent(in) iDomain )
```

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) :: iDomain*	Domain Id

Authors

Luis Samaniego

Date

Dec 2005

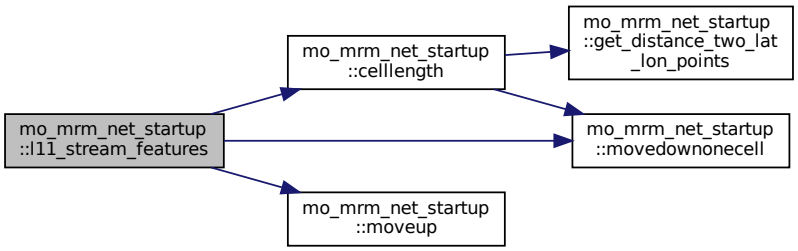
stack(nNodes, 2)

Definition at line 1227 of file mo_mrm_net_startup.f90.

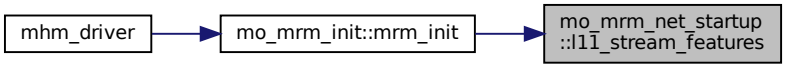
References celllength(), mo_common_variables::domainmeta, mo_common_variables::iflag_cordinate_sys, mo_common_variables::l0_elev, mo_mrm_global_variables::l0_fdir, mo_mrm_global_variables::l0_floodplain, mo_mrm_global_variables::l0_streamnet, mo_mrm_global_variables::l11_afloodplain, mo_mrm_global_variables::l11_fcol, 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:



17.42.2.13 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*	
in,out	*integer(i4) :: iRow, jCol*	

Authors

Robert Schweppe

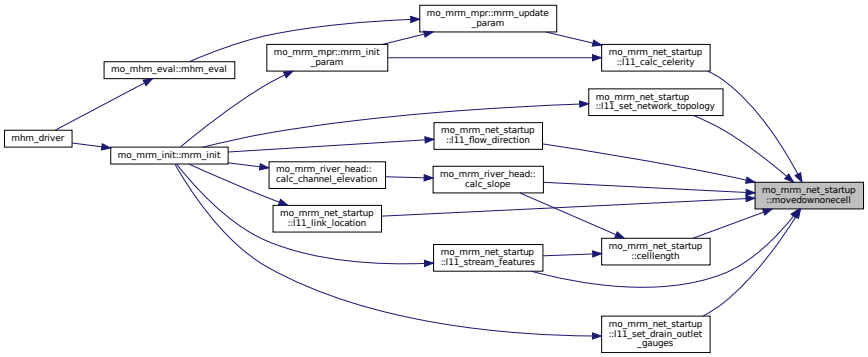
Date

Jun 2018

Definition at line 1760 of file mo_mrm_net_startup.f90.

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:



17.42.2.14 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

Definition at line 1587 of file mo_mrm_net_startup.f90.

References mo_mrm_constants::deltah.

Referenced by l11_stream_features().

Here is the caller graph for this function:

17.43 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.
- real(dp) function, public single_objective_runoff_master (parameterset, eval, arg1, arg2, arg3)

  Wrapper for objective functions optimizing agains runoff.
- subroutine, public single_objective_runoff_subprocess (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.
- real(dp) function objective_sse_boxcox (parameterset, eval)

  Objective function of sum of squared errors of transformed streamflow.
- subroutine, public extract_runoff (gaugeId, runoff, runoff_agg, runoff_obs, runoff_obs_mask)

  extracts runoff data from global variables

17.43.1 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 (32) SO: Q: SSE of boxcox-transformed streamflow

Authors

Juliane Mai

Date

Dec 2012

17.43.2 Function/Subroutine Documentation

17.43.2.1 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

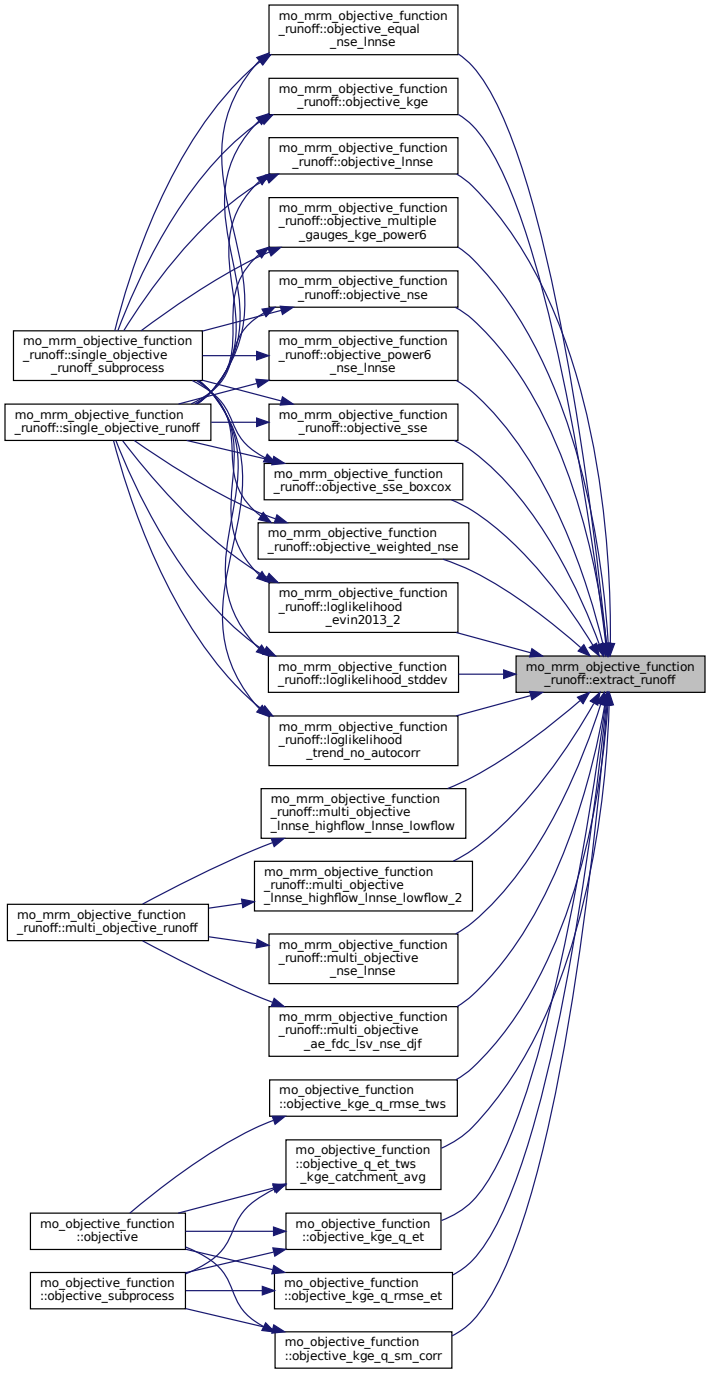
Jan 2015

Definition at line 2475 of file mo_mrm_objective_function_runoff.f90.

References mo_common_mhm_mrm_variables::evalper, mo_mrm_global_variables::gauge, 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(), mo_objective_function::objective_q_et_tws_kge_catchment_avg(), objective_sse(), objective_sse_boxcox(), and objective_weighted_nse().

Here is the caller graph for this function:



17.43.2.2 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

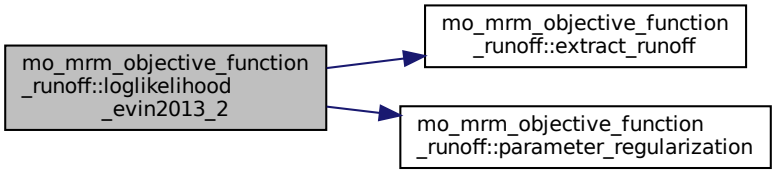
Mar 2014

Definition at line 732 of file mo_mrm_objective_function_runoff.f90.

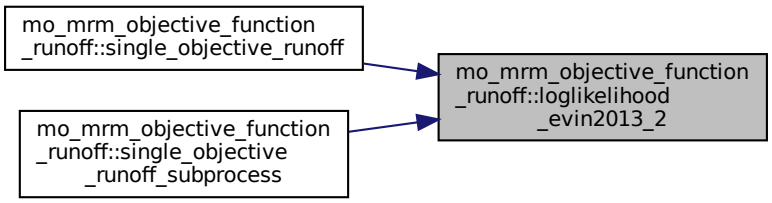
References extract_runoff(), mo_common_variables::global_parameters, and parameter_regularization().

Referenced by single_objective_runoff(), and single_objective_runoff_subprocess().

Here is the call graph for this function:



Here is the caller graph for this function:



17.43.2.3 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

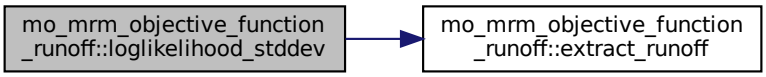
Dec 2012

Definition at line 590 of file mo_mrm_objective_function_runoff.f90.

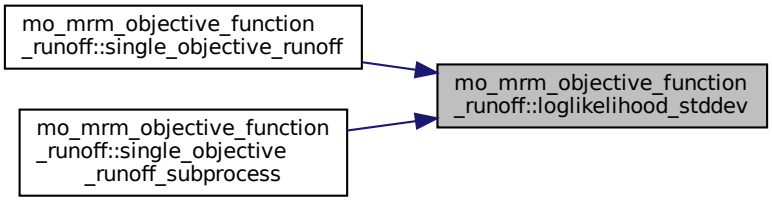
References extract_runoff().

Referenced by single_objective_runoff(), and single_objective_runoff_subprocess().

Here is the call graph for this function:



Here is the caller graph for this function:



17.43.2.4 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

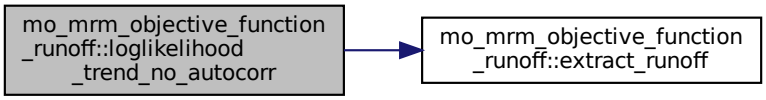
Mar 2014

Definition at line 951 of file mo_mrm_objective_function_runoff.f90.

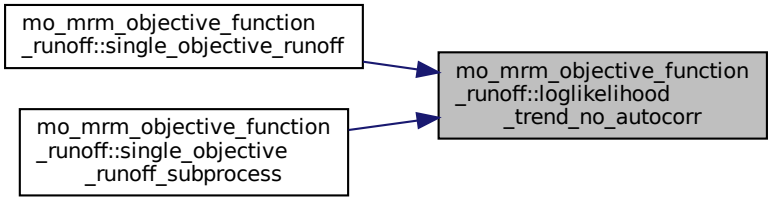
References extract_runoff().

Referenced by single_objective_runoff(), and single_objective_runoff_subprocess().

Here is the call graph for this function:



Here is the caller graph for this function:



17.43.2.5 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

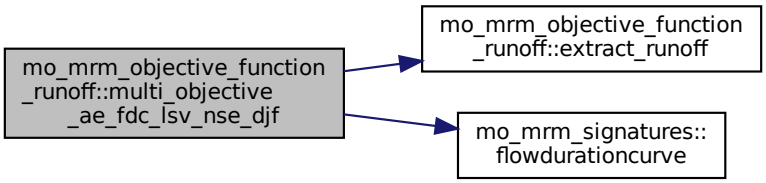
Feb 2016

Definition at line 1834 of file mo_mrm_objective_function_runoff.f90.

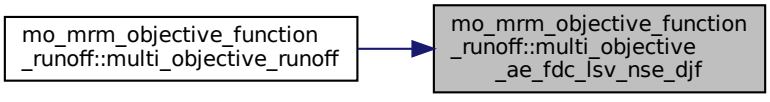
References mo_common_mhm_mrm_variables::evalper, extract_runoff(), mo_mrm_signatures::flowdurationcurve(), mo_mrm_global_variables::gauge, and mo_mrm_global_variables::nmeasperday.

Referenced by multi_objective_runoff().

Here is the call graph for this function:



Here is the caller graph for this function:



17.43.2.6 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

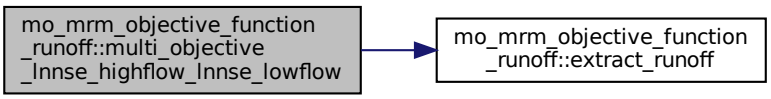
Oct 2015

Definition at line 1558 of file mo_mrm_objective_function_runoff.f90.

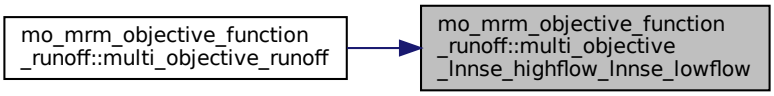
References extract_runoff().

Referenced by multi_objective_runoff().

Here is the call graph for this function:



Here is the caller graph for this function:



17.43.2.7 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

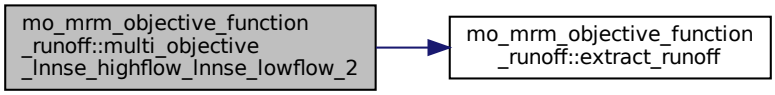
Oct 2015

Definition at line 1702 of file mo_mrm_objective_function_runoff.f90.

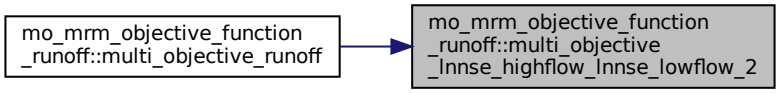
References extract_runoff().

Referenced by multi_objective_runoff().

Here is the call graph for this function:



Here is the caller graph for this function:



17.43.2.8 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

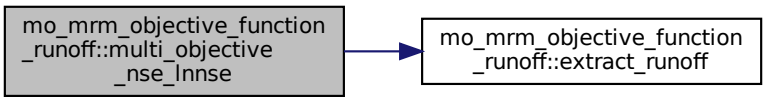
Oct 2015

Definition at line 1453 of file mo_mrm_objective_function_runoff.f90.

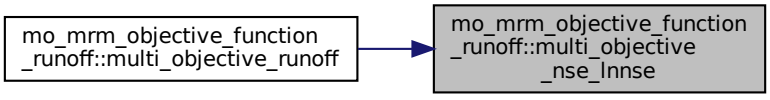
References extract_runoff().

Referenced by multi_objective_runoff().

Here is the call graph for this function:



Here is the caller graph for this function:



17.43.2.9 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

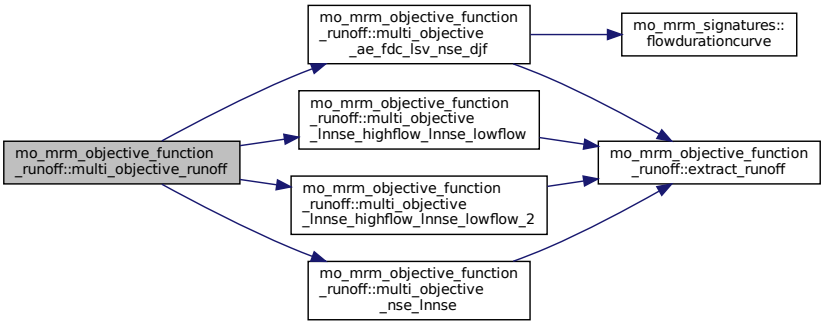
Date

Oct 2015

Definition at line 516 of file mo_mrm_objective_function_runoff.f90.

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:



17.43.2.10 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

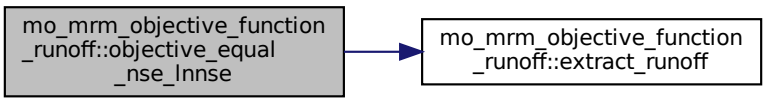
May 2013

Definition at line 1356 of file mo_mrm_objective_function_runoff.f90.

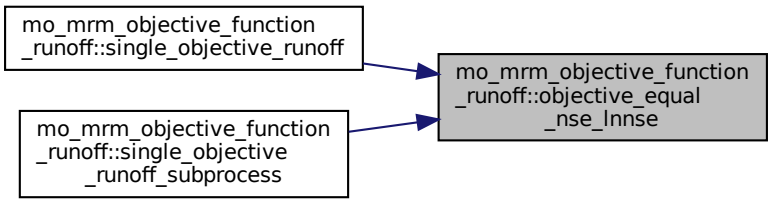
References extract_runoff().

Referenced by single_objective_runoff(), and single_objective_runoff_subprocess().

Here is the call graph for this function:



Here is the caller graph for this function:



17.43.2.11 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

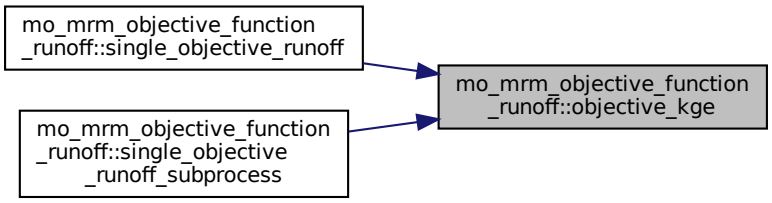
Definition at line 2094 of file mo_mrm_objective_function_runoff.f90.

References extract_runoff().

Referenced by single_objective_runoff(), and single_objective_runoff_subprocess().

Here is the call graph for this function:

Here is the caller graph for this function:



17.43.2.12 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

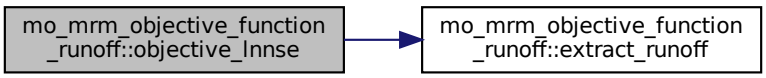
May 2013

Definition at line 1084 of file mo_mrm_objective_function_runoff.f90.

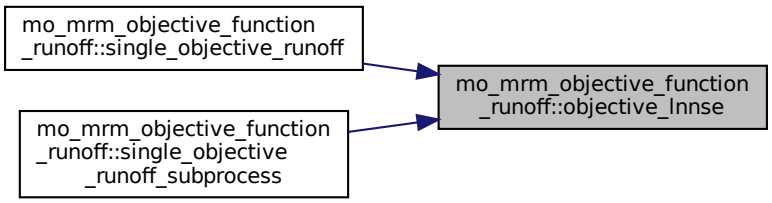
References extract_runoff().

Referenced by single_objective_runoff(), and single_objective_runoff_subprocess().

Here is the call graph for this function:



Here is the caller graph for this function:



17.43.2.13 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

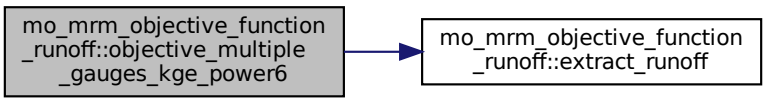
March 2015

Definition at line 2196 of file mo_mrm_objective_function_runoff.f90.

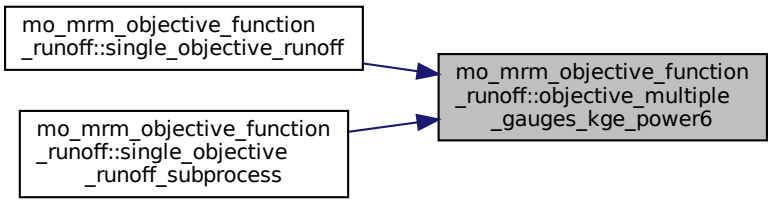
References extract_runoff().

Referenced by single_objective_runoff(), and single_objective_runoff_subprocess().

Here is the call graph for this function:



Here is the caller graph for this function:



17.43.2.14 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

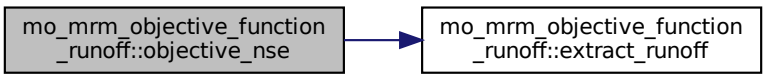
May 2013

Definition at line 1263 of file mo_mrm_objective_function_runoff.f90.

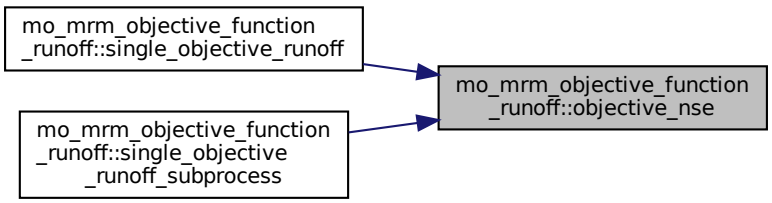
References extract_runoff().

Referenced by single_objective_runoff(), and single_objective_runoff_subprocess().

Here is the call graph for this function:



Here is the caller graph for this function:



17.43.2.15 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

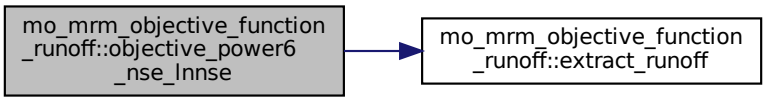
March 2014

Definition at line 1994 of file mo_mrm_objective_function_runoff.f90.

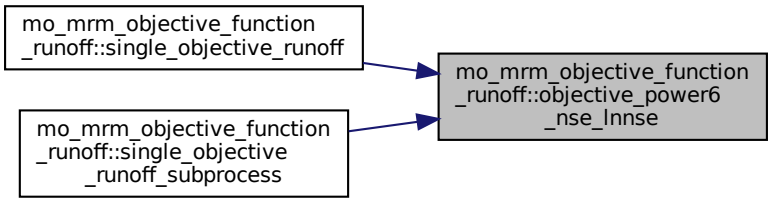
References extract_runoff().

Referenced by single_objective_runoff(), and single_objective_runoff_subprocess().

Here is the call graph for this function:



Here is the caller graph for this function:



17.43.2.16 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

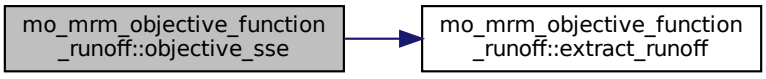
March 2014

Definition at line 1173 of file mo_mrm_objective_function_runoff.f90.

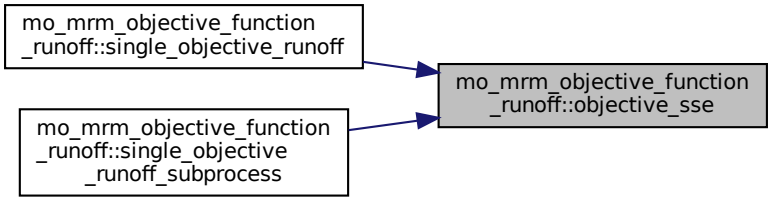
References extract_runoff().

Referenced by single_objective_runoff(), and single_objective_runoff_subprocess().

Here is the call graph for this function:



Here is the caller graph for this function:



17.43.2.17 objective_sse_boxcox()

```
real(dp) function mo_mrm_objective_function_runoff::objective_sse_boxcox (
            real(dp), dimension(:), intent(in) parameterset,
            procedure(eval_interface), intent(in), pointer eval )
```

Objective function of sum of squared errors of transformed streamflow.

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 error $tSSE$

$$tSSE = \sum_{i=1}^{N} (z(Q_{obs}(i), \lambda) - z(Q_{model}(i), \lambda))^2$$

is calculated where z is the transform and given by

$$z(x, \lambda) = \frac{x^{\lambda} - 1}{\lambda}$$

for λ unequal to zero and

$$z(x, \lambda) = logx$$

for λ equal to zero. The objective function is

$$obj\_value = tSSE$$

The observed data Q_{obs} are global in this module.

The boxcox transformation uses a parameter of 0.2, suggested by Woldemeskel et al. Hydrol Earth Syst Sci, 2018 vol. 22 (12) pp. 6257-6278. "Evaluating post-processing approaches for monthly and seasonal streamflow forecasts." https://www.hydrol-earth-syst-sci.net/22/6257/2018/

Parameters

in	*real(dp), dimension(:) :: parameterset*	
in	*procedure(eval_interface) :: eval*	

Returns

real(dp) :: objective_sse_boxcox — objective function value (which will be e.g. minimized by an optimization routine like DDS)

Authors

Stephan Thober, Dmitri Kavetski

Date

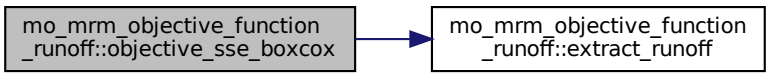
Aug 2019

Definition at line 2377 of file mo_mrm_objective_function_runoff.f90.

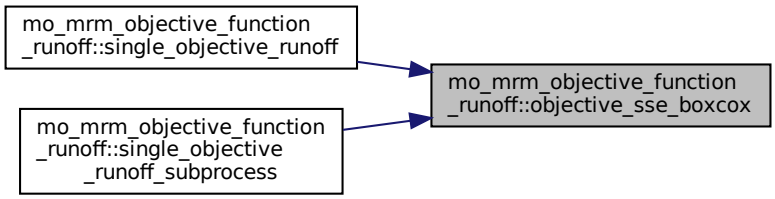
References extract_runoff().

Referenced by single_objective_runoff(), and single_objective_runoff_subprocess().

Here is the call graph for this function:



Here is the caller graph for this function:



17.43.2.18 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

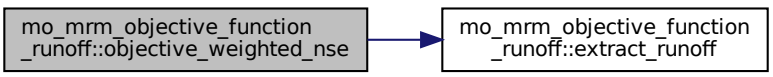
May 2018

Definition at line 2284 of file mo_mrm_objective_function_runoff.f90.

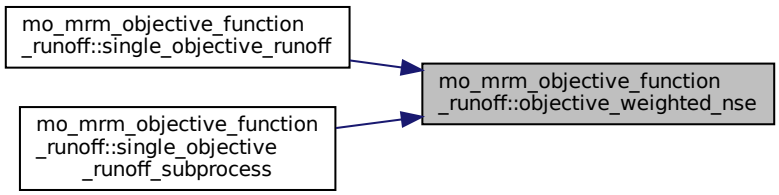
References extract_runoff().

Referenced by single_objective_runoff(), and single_objective_runoff_subprocess().

Here is the call graph for this function:



Here is the caller graph for this function:



17.43.2.19 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

Definition at line 867 of file mo_mrm_objective_function_runoff.f90.

Referenced by loglikelihood_evin2013_2().

Here is the caller graph for this function:

17.43.2.20 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

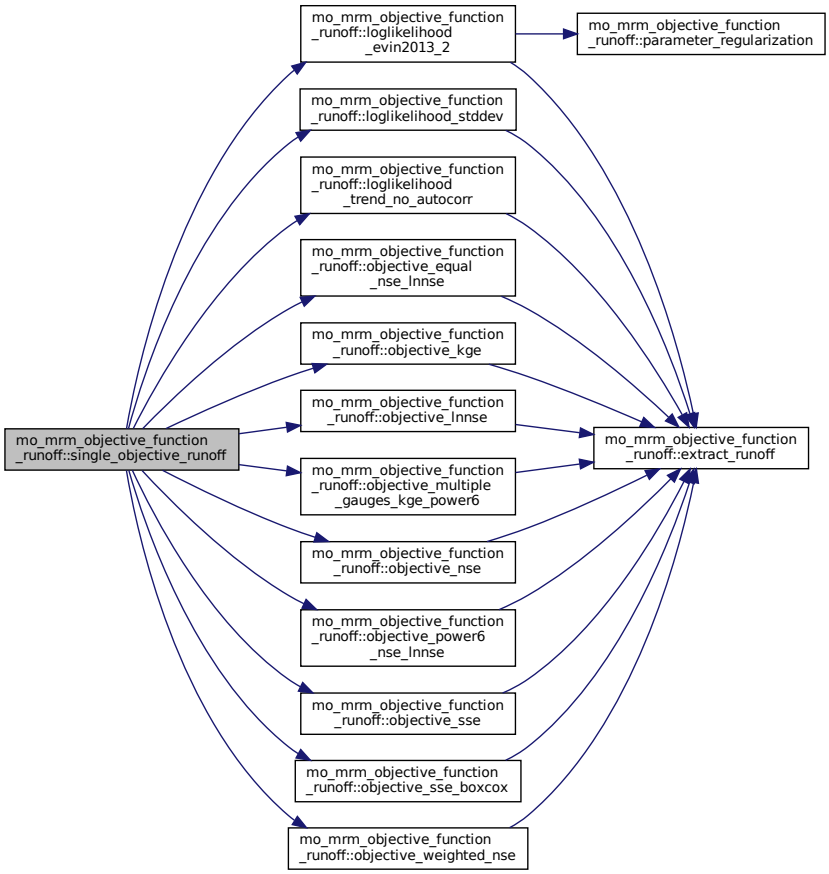
Date

Dec 2012

Definition at line 116 of file mo_mrm_objective_function_runoff.f90.

References loglikelihood_evin2013_2(), loglikelihood_stddev(), loglikelihood_trend_no_autocorr(), objective_↩ equal_nse_lnnse(), objective_kge(), objective_lnnse(), objective_multiple_gauges_kge_power6(), objective_nse(), objective_power6_nse_lnnse(), objective_sse(), objective_sse_boxcox(), objective_weighted_nse(), mo_common↩ _mhm_mrm_variables::opti_function, and mo_common_mhm_mrm_variables::opti_method.

Here is the call graph for this function:



17.43.2.21 single_objective_runoff_master()

```
real(dp) function, public mo_mrm_objective_function_runoff::single_objective_runoff_master (
            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

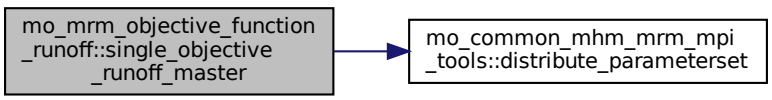
Date

Dec 2012

Definition at line 233 of file mo_mrm_objective_function_runoff.f90.

References mo_common_mhm_mrm_mpi_tools::distribute_parameterset(), mo_common_variables::domainmeta, mo_mrm_global_variables::ngaugestotal, mo_common_mhm_mrm_variables::opti_function, and mo_common_↩ mhm_mrm_variables::opti_method.

Here is the call graph for this function:



17.43.2.22 single_objective_runoff_subprocess()

```
subroutine, public mo_mrm_objective_function_runoff::single_objective_runoff_subprocess (
            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

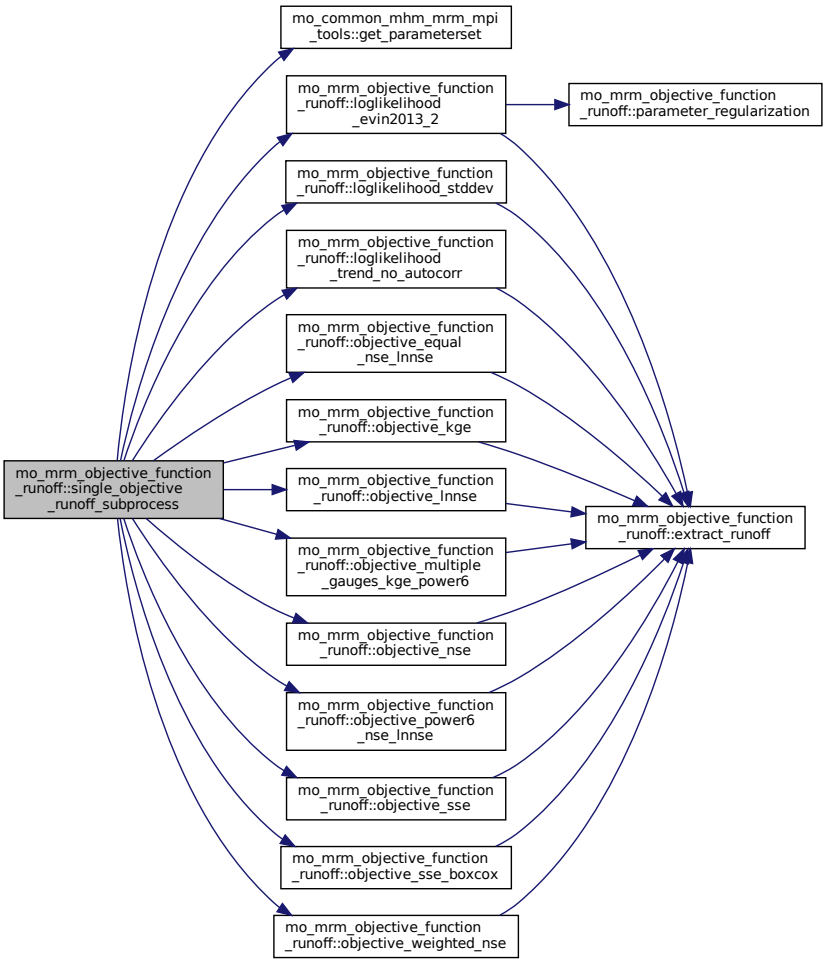
Date

Dec 2012

Definition at line 381 of file mo_mrm_objective_function_runoff.f90.

References mo_common_variables::domainmeta, mo_common_mhm_mrm_mpi_tools::get_parameterset(), loglikelihood_evin2013_2(), loglikelihood_stddev(), loglikelihood_trend_no_autocorr(), objective_equal_nse_↩lnnse(), objective_kge(), objective_lnnse(), objective_multiple_gauges_kge_power6(), objective_nse(), objective↩_power6_nse_lnnse(), objective_sse(), objective_sse_boxcox(), objective_weighted_nse(), mo_common_mhm_↩mrm_variables::opti_function, and mo_common_mhm_mrm_variables::opti_method.

Here is the call graph for this function:



17.44 mo_mrm_pre_routing Module Reference

Performs pre-processing for routing for mHM at level L11.

Functions/Subroutines

- subroutine, public l11_runoff_acc (qAll, efecArea, L1_L11_Id, L11_areaCell, L11_L1_Id, TS, map_flag, qAcc)

  total runoff accumulation at L11.
- subroutine, public add_inflow (nInflowGauges, InflowIndexList, InflowHeadwater, InflowNodeList, QInflow, q↩ Out)

  Adds inflow discharge to the runoff produced at the cell where the inflow is occurring.
- subroutine l11_e_acc (qAll, efecArea, L1_L11_Id, L11_areaCell, L11_L1_Id, TS, map_flag, qAcc)

  temperature energy accumulation at L11.
- subroutine, public calc_l1_runoff_e (fSealed_area_fraction, fast_interflow, slow_interflow, baseflow, direct_↩ runoff, temp_air, mean_temp_air, lateral_E)

  calculate lateral temperature energy from runoff components.

- subroutine, public l11_meteo_acc (meteo_all, efecArea, L1_L11_Id, L11_areaCell, L11_L1_Id, map_flag, meteo_acc)

  meteo forcing accumulation at L11 for temperature routing.

17.44.1 Detailed Description

Performs pre-processing for routing for mHM at level L11.

This module performs runoff accumulation from L1 to L11 and inflow summation.

Authors

Luis Samaniego

Date

Dec 2012

17.44.2 Function/Subroutine Documentation

17.44.2.1 add_inflow()

```
subroutine, public mo_mrm_pre_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

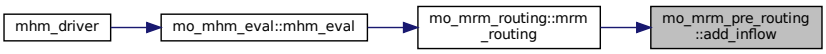
Date

Jul 2016

Definition at line 177 of file mo_mrm_pre_routing.f90.

Referenced by mo_mrm_routing::mrm_routing().

Here is the caller graph for this function:



17.44.2.2 calc_l1_runoff_e()

```
subroutine, public mo_mrm_pre_routing::calc_l1_runoff_e (
             real(dp), dimension(:), intent(in) fSealed_area_fraction,
             real(dp), dimension(:), intent(in) fast_interflow,
             real(dp), dimension(:), intent(in) slow_interflow,
             real(dp), dimension(:), intent(in) baseflow,
             real(dp), dimension(:), intent(in) direct_runoff,
             real(dp), dimension(:), intent(in) temp_air,
             real(dp), dimension(:), intent(in) mean_temp_air,
             real(dp), dimension(:), intent(inout) lateral_E )
```

calculate lateral temperature energy from runoff components.

calculate lateral temperature energy from runoff components.

Parameters

in	*REAL(dp) :: fSealed_area_fraction*	sealed area fraction [1]
in	*REAL(dp) :: fast_interflow*	q_0 Fast runoff component [mm TS-1]
in	*REAL(dp) :: slow_interflow*	q_1 Slow runoff component [mm TS-1]
in	*REAL(dp) :: baseflow*	q_2 Baseflow [mm TS-1]
in	*REAL(dp) :: direct_runoff*	q_D Direct runoff from impervious areas [mm TS-1]
in	*REAL(dp) :: temp_air*	air temperature [K]
in	*REAL(dp) :: mean_temp_air*	annual mean air temperature [K]
out	*REAL(dp) :: lateral_E*	E_T Generated runoff [K mm TS-1]

Authors

Sebastian Mueller

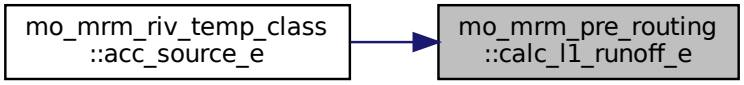
Date

Jun 2020

Definition at line 346 of file mo_mrm_pre_routing.f90.

Referenced by mo_mrm_riv_temp_class::acc_source_e().

Here is the caller graph for this function:



17.44.2.3 l11_e_acc()

```
subroutine mo_mrm_pre_routing::l11_e_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 )
```

temperature energy accumulation at L11.

Upscales energy 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 K TS-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 K s-1]

Authors

Luis Samaniego

Date

Jan 2013

Definition at line 253 of file mo_mrm_pre_routing.f90.

References mo_common_constants::nodata_dp.

17.44.2.4 l11_meteo_acc()

```
subroutine, public mo_mrm_pre_routing::l11_meteo_acc (
             real(dp), dimension(:), intent(in) meteo_all,
             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,
             logical, intent(in) map_flag,
             real(dp), dimension(:), intent(out) meteo_acc )
```

meteo forcing accumulation at L11 for temperature routing.

Upscales meteo forcing in space from L1 to L11 if routing resolution is higher than hydrology resolution (map_flag equals .true.) or downscales meteo forcing from L1 to L11 if routing resolution is lower than hydrology resolution.

Parameters

in	*real(dp), dimension(:) :: meteo_all*	meteo forcing
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	*logical :: map_flag*	Flag indicating whether routing resolution is higher than hydrologic one
out	*real(dp), dimension(:) :: meteo_acc*	aggregated meteo forcing

Authors

Sebastian Mueller

Date

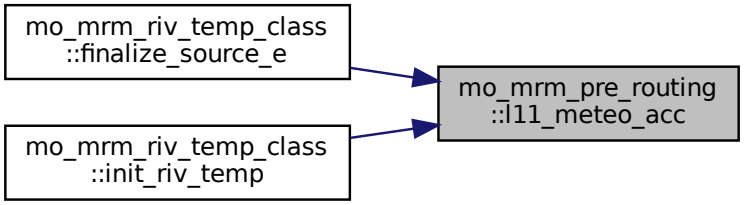
Jul 2020

Definition at line 420 of file mo_mrm_pre_routing.f90.

References mo_common_constants::nodata_dp.

Referenced by mo_mrm_riv_temp_class::finalize_source_e(), and mo_mrm_riv_temp_class::init_riv_temp().

Here is the caller graph for this function:



17.44.2.5 l11_runoff_acc()

```
subroutine, public mo_mrm_pre_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 TS-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

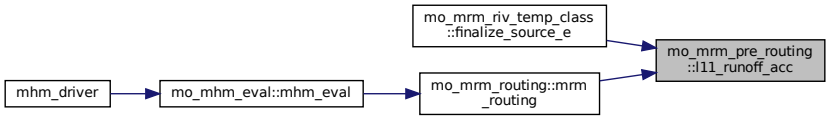
Jan 2013

Definition at line 75 of file mo_mrm_pre_routing.f90.

References mo_common_constants::nodata_dp.

Referenced by mo_mrm_riv_temp_class::finalize_source_e(), and mo_mrm_routing::mrm_routing().

Here is the caller graph for this function:



17.45 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.

17.45.1 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

17.45.2 Function/Subroutine Documentation

17.45.2.1 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

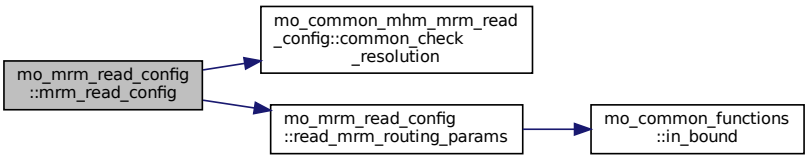
Aug 2015

Definition at line 56 of file mo_mrm_read_config.f90.

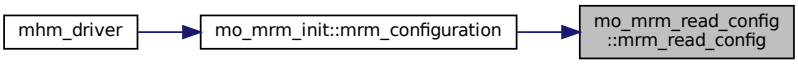
References mo_common_variables::alma_convention, mo_common_mhm_mrm_read_config::common_check_resolution(), mo_mrm_global_variables::dirbankfullrunoff, mo_mrm_global_variables::dirgauges, mo_mrm_global_variables::dirtotalrunoff, mo_mrm_global_variables::domain_mrm, mo_common_variables::domainmeta, mo_mrm_file::file_defoutput, 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::maxnodomains, mo_mrm_constants::maxnogauges, mo_mrm_global_variables::ngaugeslocal, mo_mrm_global_variables::ngaugestotal, mo_mrm_global_variables::ninflowgaugestotal, mo_common_constants::nodata_i4, mo_mrm_global_variables::output_deflate_level_mrm, mo_mrm_global_variables::output_double_precision_mrm, mo_mrm_global_variables::outputflxstate_mrm, 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_configuration().

Here is the call graph for this function:



Here is the caller graph for this function:



17.45.2.2 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

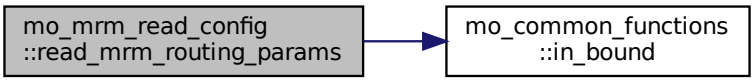
Jun 2018

Definition at line 429 of file mo_mrm_read_config.f90.

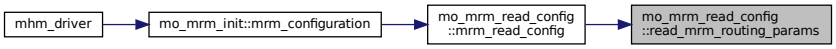
References mo_common_variables::global_parameters, mo_common_variables::global_parameters_name, mo_↩ common_functions::in_bound(), mo_common_constants::ncolpars, and mo_common_variables::processmatrix.

Referenced by mrm_read_config().

Here is the call graph for this function:



Here is the caller graph for this function:



17.46 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 (iDomain)
  read simulated runoff that is to be routed
- subroutine, public mrm_read_bankfull_runoff (iDomain)
- subroutine rotate_fdir_variable (x)
  TODO: add description.

17.46.1 Detailed Description

This module contains all routines to read mRM data from file.

TODO: add description

Authors

Stephan Thober

Date

Aug 2015

17.46.2 Function/Subroutine Documentation

17.46.2.1 mrm_read_bankfull_runoff()

```
subroutine, public mo_mrm_read_data::mrm_read_bankfull_runoff (
             integer(i4), intent(in) iDomain )
```

Parameters

in	*idomain*	

Note

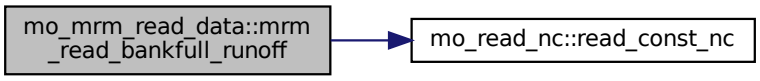
The file read in must contain a double precision float variable with the name "Q_bkfl".

Definition at line 432 of file mo_mrm_read_data.f90.

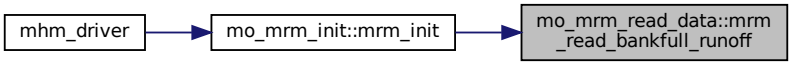
References mo_mrm_global_variables::dirbankfullrunoff, mo_mrm_global_variables::level11, and mo_read_nc↩ ::read_const_nc().

Referenced by mo_mrm_init::mrm_init().

Here is the call graph for this function:



Here is the caller graph for this function:



17.46.2.2 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

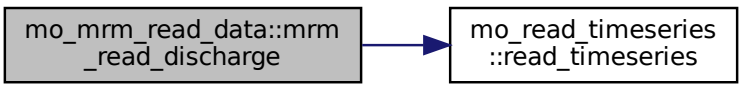
Aug 2015

Definition at line 256 of file mo_mrm_read_data.f90.

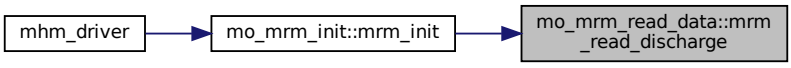
References mo_common_variables::domainmeta, mo_common_mhm_mrm_variables::evalper, mo_mrm_global_variables::gauge, mo_mrm_global_variables::inflowgauge, mo_mrm_global_variables::mrm_runoff, mo_mrm_global_variables::ngaugeslocal, mo_mrm_global_variables::ninflowgaugestotal, mo_mrm_global_variables::nmeasperday, 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_mrm_global_variables::riv_temp_pcs, 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:



17.46.2.3 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

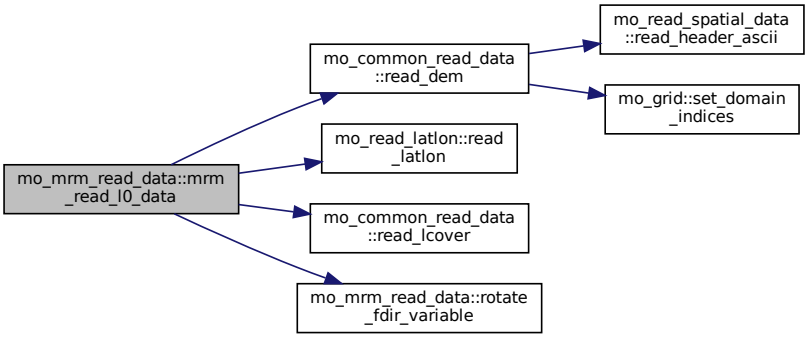
Aug 2015

Definition at line 50 of file mo_mrm_read_data.f90.

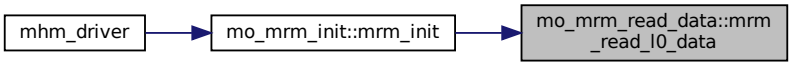
References mo_common_variables::dirmorpho, mo_mrm_global_variables::domain_mrm, mo_common_↩ variables::domainmeta, mo_mrm_file::file_facc, mo_mrm_file::file_fdir, mo_mrm_file::file_gaugeloc, mo_mpr_↩ file::file_slope, mo_mrm_global_variables::l0_facc, mo_mrm_global_variables::l0_fdir, mo_mrm_global_variables↩ ::l0_gaugeloc, mo_mrm_global_variables::l0_inflowgaugeloc, mo_common_variables::l0_lcover, mo_common_↩ variables::level0, 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:



17.46.2.4 mrm_read_total_runoff()

```
subroutine, public mo_mrm_read_data::mrm_read_total_runoff (
            integer(i4), intent(in) iDomain )
```

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) :: iDomain*	domain id

Authors

Stephan Thober

Date

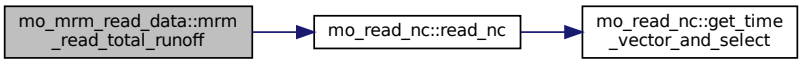
Sep 2015

Definition at line 373 of file mo_mrm_read_data.f90.

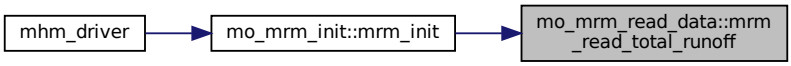
References mo_common_variables::alma_convention, mo_mrm_global_variables::dirtotalrunoff, mo_mrm_↩ global_variables::filenametotalrunoff, mo_mrm_global_variables::l1_total_runoff_in, mo_common_variables↩ ::level1, mo_common_constants::nodata_dp, mo_read_nc::read_nc(), mo_common_mhm_mrm_variables::simper, 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:



17.46.2.5 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

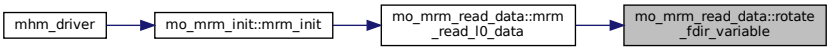
Jun 2018

Definition at line 533 of file mo_mrm_read_data.f90.

References mo_common_constants::nodata_i4.

Referenced by mrm_read_l0_data().

Here is the caller graph for this function:



17.47 mo_mrm_restart Module Reference

Restart routines.

Functions/Subroutines

- subroutine, public mrm_write_restart (iDomain, domainID, OutFile)

  write routing states and configuration
- subroutine, public mrm_read_restart_states (iDomain, domainID, InFile)

  read routing states
- subroutine, public mrm_read_restart_config (iDomain, domainID, InFile)

  reads Level 11 configuration from a restart directory

17.47.1 Detailed Description

Restart routines.

This module contains the subroutines for reading and writing routing related variables to file.

Authors

Stephan Thober

Date

Aug 2015

17.47.2 Function/Subroutine Documentation

17.47.2.1 mrm_read_restart_config()

```
subroutine, public mo_mrm_restart::mrm_read_restart_config (
            integer(i4), intent(in) iDomain,
            integer(i4), intent(in) domainID,
            character(256), intent(in) InFile )
```

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) :: iDomain*	number of Domain
in	*character(256) :: InFile*	Input Path including trailing slash

Authors

Stephan Thober

Date

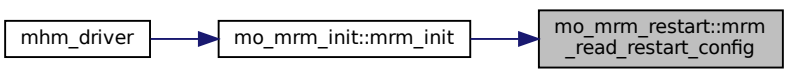
Apr 2013

Definition at line 603 of file mo_mrm_restart.f90.

References mo_mrm_global_variables::domain_mrm, mo_common_variables::domainmeta, 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_common_constants::nodata_dp, and mo_common_variables::processmatrix.

Referenced by mo_mrm_init::mrm_init().

Here is the caller graph for this function:



17.47.2.2 mrm_read_restart_states()

```
subroutine, public mo_mrm_restart::mrm_read_restart_states (
            integer(i4), intent(in) iDomain,
            integer(i4), intent(in) domainID,
            character(256), intent(in) InFile )
```

read routing states

This subroutine reads the routing states from mRM_states_<domain_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) :: iDomain*	number of domains
in	*character(256) :: InFile*	Input Path including trailing slash

Authors

Stephan Thober

Date

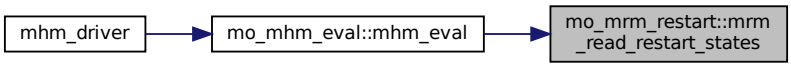
Sep 2015

Definition at line 460 of file mo_mrm_restart.f90.

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().

Here is the caller graph for this function:



17.47.2.3 mrm_write_restart()

```
subroutine, public mo_mrm_restart::mrm_write_restart (
            integer(i4), intent(in) iDomain,
            integer(i4), intent(in) domainID,
            character(256), dimension(:), intent(in) OutFile )
```

write routing states and configuration

write configuration and state variables to a given restart directory.

Parameters

in	*integer(i4) :: iDomain*	number of domain
in	*character(256), dimension(:) :: OutFile*	list of Output paths per Domain

Authors

Stephan Thober

Date

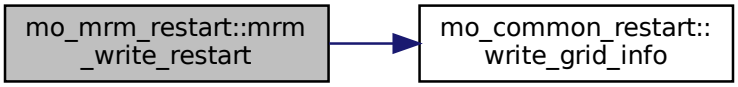
Aug 2015

Definition at line 55 of file mo_mrm_restart.f90.

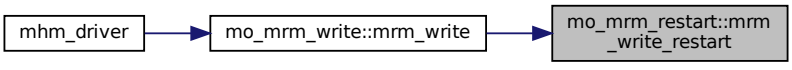
References mo_mrm_global_variables::domain_mrm, mo_common_variables::domainmeta, mo_mrm_global_variables::l0_facc, mo_mrm_global_variables::l0_fdir, mo_mpr_global_variables::l0_slope, mo_mrm_global_variables::l0_streamnet, mo_mrm_global_variables::l11_afloodplain, mo_mrm_global_variables::l11_c1, mo_mrm_global_variables::l11_c2, 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_constants::landcoverperiodsvarname, mo_common_variables::lc_year_end, mo_common_variables::lc_year_start, mo_common_variables::level0, mo_common_variables::level1, mo_mrm_global_variables::level11, 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:



17.48 mo_mrm_riv_temp_class Module Reference

Class for the river temperature calculations.

Data Types

- module riv_temp_type

  This is a container to define the river temperature routing in the current time step.

Functions/Subroutines

- subroutine config (self, file_namelist, unamelist, file_namelist_param, unamelist_param)

  configure the riv_temp_type class from the mhm namelist
- subroutine init (self, nCells)

  initalize the riv_temp_type class for the current domain
- subroutine init_area (self, iDomain, L11_netPerm, L11_fromN, L11_length, nLinks, nCells, nrows, ncols, L11_mask)

  initialize the river area of riv_temp_type class for the current domain
- subroutine init_riv_temp (self, time, ntimesteps_day, temp_air, read_meteo_weights, temp_weights, fday_↩ temp, fnight_temp, efecarea, L1_L11_Id, L11_areacell, L11_L1_Id, map_flag)

  initialize the river temperature of riv_temp_type class for the current domain
- subroutine reset_timestep (self)

  reset riv_temp_type class for next timestep
- subroutine alloc_lateral (self, nCells)

  allocate lateral temp components of riv_temp_type class for current domain
- subroutine dealloc_lateral (self)

  deallocate lateral temp components of riv_temp_type
- subroutine acc_source_e (self, time, ntimesteps_day, fSealed_area_fraction, fast_interflow, slow_interflow, baseflow, direct_runoff, temp_air, mean_temp_air, ssrd_day, strd_day, read_meteo_weights, temp_weights, fday_temp, fnight_temp, fday_ssrd, fnight_ssrd, fday_strd, fnight_strd)

  accumulate energy sources of riv_temp_type
- subroutine finalize_source_e (self, efecarea, L1_L11_Id, L11_areacell, L11_L1_Id, timestep, map_flag)

  finalize energy sources of riv_temp_type
- real(dp) function get_lrad_out (self, riv_temp)

  get outgoing longwave radiation of riv_temp_type
- real(dp) function get_lat_heat (self, air_temp, netrad)

  latent heat flux of riv_temp_type
- real(dp) function get_sens_heat (self, air_temp, riv_temp)

  sensible heat flux of riv_temp_type
- real(dp) function get_e_io (self, riv_temp, cell)

  get complete energy source of riv_temp_type at given cell
- subroutine l11_routing_e (self, nLinks, netPerm, netLink_fromN, netLink_toN, netLink_C1, netLink_C2, n↩ InflowGauges, InflowHeadwater, InflowNodeList, L11_qTR, L11_Qmod)

  execute the temperature routing of riv_temp_type
- subroutine init_iter (self)

  initialize iterative solver of riv_temp_type
- subroutine next_iter (self, T_est, T_rout)

  execute next iteration with iterative solver of riv_temp_type

17.48.1 Detailed Description

Class for the river temperature calculations.

Warning

This feature is still experimental!

Version

0.1

Authors

Sebastian Mueller

Date

Sep 2020

17.48.2 Function/Subroutine Documentation

17.48.2.1 acc_source_e()

```
subroutine mo_mrm_riv_temp_class::acc_source_e (
            class(riv_temp_type), intent(inout) self,
            real(dp), intent(in) time,
            real(dp), intent(in) ntimesteps_day,
            real(dp), dimension(:), intent(in) fSealed_area_fraction,
            real(dp), dimension(:), intent(in) fast_interflow,
            real(dp), dimension(:), intent(in) slow_interflow,
            real(dp), dimension(:), intent(in) baseflow,
            real(dp), dimension(:), intent(in) direct_runoff,
            real(dp), dimension(:), intent(in) temp_air,
            real(dp), dimension(:), intent(in) mean_temp_air,
            real(dp), dimension(:), intent(in) ssrd_day,
            real(dp), dimension(:), intent(in) strd_day,
            logical, intent(in) read_meteo_weights,
            real(dp), dimension(:, :, :), intent(in) temp_weights,
            real(dp), dimension(:), intent(in) fday_temp,
            real(dp), dimension(:), intent(in) fnight_temp,
            real(dp), dimension(:), intent(in) fday_ssrd,
            real(dp), dimension(:), intent(in) fnight_ssrd,
            real(dp), dimension(:), intent(in) fday_strd,
            real(dp), dimension(:), intent(in) fnight_strd )  [private]
```

accumulate energy sources of riv_temp_type

Parameters

in	*time*	current decimal Julian day
in	*ntimesteps_day*	number of time intervals per day, transformed in dp
in	*fsealed_area_fraction*	sealed area fraction [1]
in	*fast_interflow*	q_0 Fast runoff component [mm TS-1]

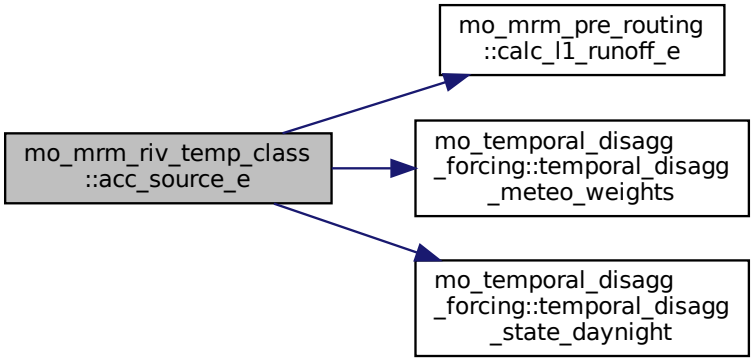
Parameters

in	*slow_interflow*	q_1 Slow runoff component [mm TS-1]
in	*baseflow*	q_2 Baseflow [mm TS-1]
in	*direct_runoff*	q_D Direct runoff from impervious areas [mm TS-1]
in	*temp_air*	air temperature [K]
in	*mean_temp_air*	annual mean air temperature [K]
in	*ssrd_day*	Daily mean short radiation
in	*strd_day*	Daily mean longwave radiation
in	*read_meteo_weights*	flag whether weights for tavg and pet have read and should be used
in	*temp_weights*	multiplicative weights for temperature (deg K)
in	*fday_temp*	[-] day factor mean temp
in	*fnight_temp*	[-] night factor mean temp
in	*fday_ssrd*	Daytime fraction of ssrd
in	*fnight_ssrd*	Nighttime fraction of ssrd
in	*fday_strd*	Daytime fraction of strd
in	*fnight_strd*	Nighttime fraction of strd

Definition at line 441 of file mo_mrm_riv_temp_class.f90.

References mo_mrm_pre_routing::calc_l1_runoff_e(), mo_temporal_disagg_forcing::temporal_disagg_meteo_↩ weights(), and mo_temporal_disagg_forcing::temporal_disagg_state_daynight().

Here is the call graph for this function:



17.48.2.2 alloc_lateral()

```
subroutine mo_mrm_riv_temp_class::alloc_lateral (
            class(riv_temp_type), intent(inout) self,
            integer(i4), intent(in) nCells )  [private]
```

allocate lateral temp components of riv_temp_type class for current domain

Parameters

in	*ncells*	number of cells for the current domain

Definition at line 407 of file mo_mrm_riv_temp_class.f90.

17.48.2.3 config()

```
subroutine mo_mrm_riv_temp_class::config (
            class(riv_temp_type), intent(inout) self,
            character(*), intent(in) file_namelist,
            integer, intent(in) unamelist,
            character(*), intent(in) file_namelist_param,
            integer, intent(in) unamelist_param )
```

configure the riv_temp_type class from the mhm namelist

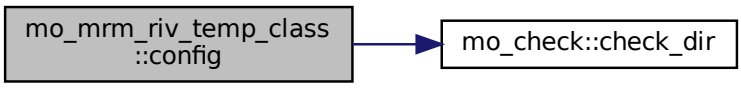
Parameters

in	*file_namelist*	mhm namelist file
in	*file_namelist_param*	mhm parameter namelist file

Definition at line 117 of file mo_mrm_riv_temp_class.f90.

References mo_check::check_dir(), mo_common_variables::domainmeta, mo_common_constants::maxnodomains, and mo_common_constants::nodata_i4.

Here is the call graph for this function:



17.48.2.4 dealloc_lateral()

```
subroutine mo_mrm_riv_temp_class::dealloc_lateral (
            class(riv_temp_type), intent(inout) self ) [private]
```

deallocate lateral temp components of riv_temp_type

Definition at line 426 of file mo_mrm_riv_temp_class.f90.

17.48.2.5 finalize_source_e()

```
subroutine mo_mrm_riv_temp_class::finalize_source_e (
```

```
class(riv_temp_type), intent(inout) self,
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) timestep,
logical, intent(in) map_flag )
```

finalize energy sources of riv_temp_type

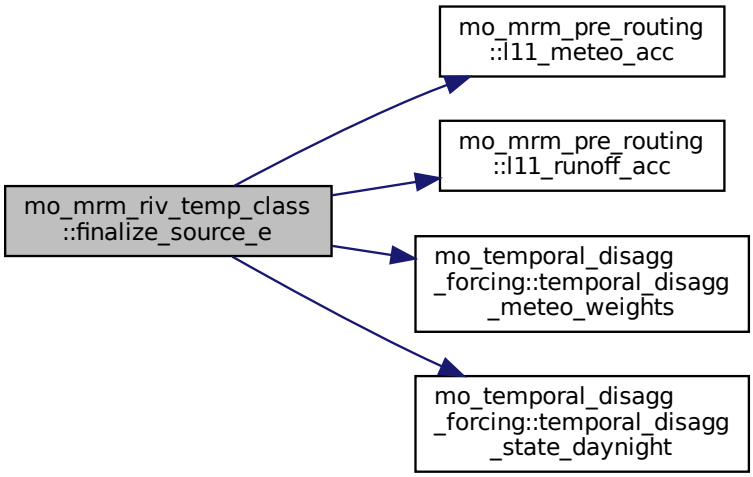
Parameters

in	*efecarea*	effective area in [km2] at Level 1
in	*l1_l11_id*	L11 Ids mapped on L1
in	*l11_areacell*	effective area in [km2] at Level 11
in	*l11_l1_id*	L1 Ids mapped on L11
in	*timestep*	simulation timestep in [h]
in	*map_flag*	Flag indicating whether routing resolution is higher than hydrologic one

Definition at line 543 of file mo_mrm_riv_temp_class.f90.

References mo_mhm_constants::h2odens, mo_mrm_pre_routing::l11_meteo_acc(), mo_mrm_pre_routing::l11_runoff_acc(), mo_temporal_disagg_forcing::temporal_disagg_meteo_weights(), and mo_temporal_disagg_forcing::temporal_disagg_state_daynight().

Here is the call graph for this function:



17.48.2.6 get_e_io()

```
real(dp) function mo_mrm_riv_temp_class::get_e_io (
            class(riv_temp_type), intent(in) self,
            real(dp), intent(in) riv_temp,
            integer(i4), intent(in) cell )  [private]
```

get complete energy source of riv_temp_type at given cell

Returns

energy IO

Parameters

in	*riv_temp*	given river temperature in K to calculate heat fluxes
in	*cell*	cell index in the current domain

net radiation calc from short and longwave radiation in/out

Definition at line 667 of file mo_mrm_riv_temp_class.f90.

References mo_mhm_constants::h2odens.

17.48.2.7 get_lat_heat()

```
real(dp) function mo_mrm_riv_temp_class::get_lat_heat (
            class(riv_temp_type), intent(in) self,
            real(dp), intent(in) air_temp,
            real(dp), intent(in) netrad )
```

latent heat flux of riv_temp_type

Returns

latent heat flux

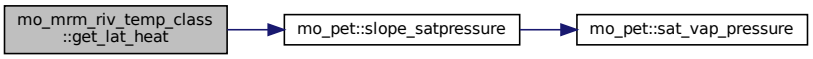
Parameters

in	*air_temp*	air temperature in deg C
in	*netrad*	net radiation in W $*$ m-2

Definition at line 628 of file mo_mrm_riv_temp_class.f90.

References mo_pet::slope_satpressure().

Here is the call graph for this function:



17.48.2.8 get_lrad_out()

```
real(dp) function mo_mrm_riv_temp_class::get_lrad_out (
            class(riv_temp_type), intent(in) self,
            real(dp), intent(in) riv_temp )
```

get outgoing longwave radiation of riv_temp_type

Returns

outgoing longwave radiation

Parameters

in	*riv_temp*	river temperature in K

Definition at line 611 of file mo_mrm_riv_temp_class.f90.

17.48.2.9 get_sens_heat()

```
real(dp) function mo_mrm_riv_temp_class::get_sens_heat (
            class(riv_temp_type), intent(in) self,
            real(dp), intent(in) air_temp,
            real(dp), intent(in) riv_temp )
```

sensible heat flux of riv_temp_type

Returns

sensible heat flux

Parameters

in	*air_temp*	air temperature in [deg C]
in	*riv_temp*	river temperature in [deg C]

Definition at line 652 of file mo_mrm_riv_temp_class.f90.

17.48.2.10 init()

```
subroutine mo_mrm_riv_temp_class::init (
            class(riv_temp_type), intent(inout) self,
            integer(i4), intent(in) nCells )
```

initalize the riv_temp_type class for the current domain

Parameters

in	*ncells*	number of cells for the current domain

Definition at line 215 of file mo_mrm_riv_temp_class.f90.

References mo_mrm_constants::nroutingstates.

17.48.2.11 init_area()

```
subroutine mo_mrm_riv_temp_class::init_area (
             class(riv_temp_type), intent(inout) self,
             integer(i4), intent(in) iDomain,
             integer(i4), dimension(:), intent(in) L11_netPerm,
             integer(i4), dimension(:), intent(in) L11_fromN,
             real(dp), dimension(:), intent(in) L11_length,
             integer(i4), intent(in) nLinks,
             integer(i4), intent(in) nCells,
             integer(i4), intent(in) nrows,
             integer(i4), intent(in) ncols,
             logical, dimension(:, :), intent(in) L11_mask )
```

initialize the river area of riv_temp_type class for the current domain

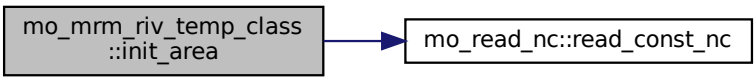
Parameters

in	*idomain*	Domain ID
in	*l11_netperm*	L11 routing order
in	*l11_fromn*	L11 source grid cell order
in	*l11_length*	L11 link length
in	*nlinks*	number of L11 links in the current domain
in	*ncells*	number of L11 cells of the current domain
in	*ncols*	Number of columns
in	*nrows*	Number of rows
in	*l11_mask*	the mask for valid cells in the original grid (nrows, ncols)

Definition at line 258 of file mo_mrm_riv_temp_class.f90.

References mo_read_nc::read_const_nc().

Here is the call graph for this function:



17.48.2.12 init_iter()

```
subroutine mo_mrm_riv_temp_class::init_iter (
             class(riv_temp_type), intent(inout) self )
```

initialize iterative solver of riv_temp_type

Definition at line 812 of file mo_mrm_riv_temp_class.f90.

17.48.2.13 init_riv_temp()

```
subroutine mo_mrm_riv_temp_class::init_riv_temp (
             class(riv_temp_type), intent(inout) self,
             real(dp), intent(in) time,
             real(dp), intent(in) ntimesteps_day,
             real(dp), dimension(:), intent(in) temp_air,
             logical, intent(in) read_meteo_weights,
             real(dp), dimension(:, :, :), intent(in) temp_weights,
             real(dp), dimension(:), intent(in) fday_temp,
             real(dp), dimension(:), intent(in) fnight_temp,
             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,
             logical, intent(in) map_flag )
```

initialize the river temperature of riv_temp_type class for the current domain

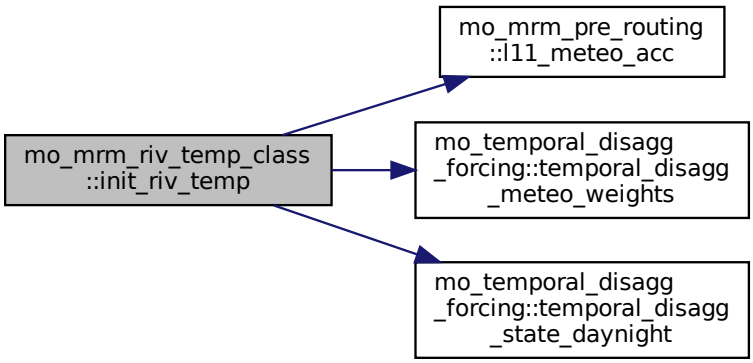
Parameters

in	*time*	current decimal Julian day
in	*ntimesteps_day*	number of time intervals per day, transformed in dp
in	*temp_air*	air temperature [K]
in	*read_meteo_weights*	flag whether weights for tavg and pet have read and should be used
in	*temp_weights*	multiplicative weights for temperature (deg K)
in	*fday_temp*	[-] day factor mean temp
in	*fnight_temp*	[-] night factor mean temp
in	*efecarea*	effective area in [km2] at Level 1
in	*l1_l11_id*	L11 Ids mapped on L1
in	*l11_areacell*	effective area in [km2] at Level 11
in	*l11_l1_id*	L1 Ids mapped on L11
in	*map_flag*	Flag indicating whether routing resolution is higher than hydrologic one

Definition at line 322 of file mo_mrm_riv_temp_class.f90.

References mo_mrm_pre_routing::l11_meteo_acc(), mo_temporal_disagg_forcing::temporal_disagg_meteo_↩ weights(), and mo_temporal_disagg_forcing::temporal_disagg_state_daynight().

Here is the call graph for this function:



17.48.2.14 l11_routing_e()

```
subroutine mo_mrm_riv_temp_class::l11_routing_e (
             class(riv_temp_type), intent(inout) self,
             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,
             integer(i4), intent(in) nInflowGauges,
             logical, dimension(:), intent(in) InflowHeadwater,
             integer(i4), dimension(:), intent(in) InflowNodeList,
             real(dp), dimension(:), intent(in) L11_qTR,
             real(dp), dimension(:), intent(in) L11_Qmod )
```

execute the temperature routing of riv_temp_type

Parameters

in	*nlinks*	number of stream segment (reaches)
in	*netperm*	routing order of a given domain (permutation)
in	*netlink_fromn*	from node
in	*netlink_ton*	to node
in	*netlink_c1*	routing parameter C1 ([7] p. 25-41)
in	*netlink_c2*	routing parameters C2 (id)
in	*ninflowgauges*	[-] number of inflow points
in	*inflowheadwater*	[-] if to consider headwater cells of inflow gauge
in	*inflownodelist*	[-] L11 ID of inflow points
in	*l11_qtr*	[m3 s-1] Transformed outflow leaving node I at current timestep(Muskingum)
in	*l11_qmod*	[m3 s-1] Simulated routed discharge

Definition at line 696 of file mo_mrm_riv_temp_class.f90.

17.48.2.15 next_iter()

```
subroutine mo_mrm_riv_temp_class::next_iter (
             class(riv_temp_type), intent(inout) self,
             real(dp), intent(inout) T_est,
             real(dp), intent(in) T_rout )  [private]
```

execute next iteration with iterative solver of riv_temp_type

Parameters

in,out	*t_est*	estimated river temperature
in	*t_rout*	calculated (routed) river temperature

Definition at line 826 of file mo_mrm_riv_temp_class.f90.

17.48.2.16 reset_timestep()

```
subroutine mo_mrm_riv_temp_class::reset_timestep (
             class(riv_temp_type), intent(inout) self )
```

reset riv_temp_type class for next timestep

Definition at line 391 of file mo_mrm_riv_temp_class.f90.

17.49 mo_mrm_river_head Module Reference

Functions/Subroutines

- subroutine, public init_masked_zeros_l0 (iDomain, data)
- subroutine, private reset_sum (iDomain, data)
- subroutine, public calc_channel_elevation ()
- subroutine, public calc_river_head (iDomain, L11_Qmod, river_head)
- real(dp) function calc_slope (iDomain, elev0, fDir0, i, j)
- subroutine, public avg_and_write_timestep (iDomain, timestep, data)
- subroutine, public create_output (iDomain, 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

17.49.1 Function/Subroutine Documentation

17.49.1.1 avg_and_write_timestep()

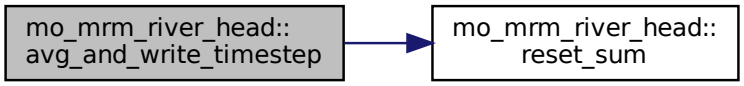
```
subroutine, public mo_mrm_river_head::avg_and_write_timestep (
             integer(i4), intent(in) iDomain,
             integer(i4), intent(in) timestep,
             real(dp), dimension(:), intent(inout) data )
```

Definition at line 188 of file mo_mrm_river_head.f90.

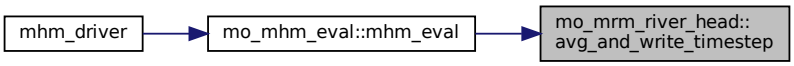
References mo_common_variables::level0, nc_riverhead, nc_time, mo_common_constants::nodata_dp, reset_sum(), sum_counter, and time_counter.

Referenced by mo_mhm_eval::mhm_eval().

Here is the call graph for this function:



Here is the caller graph for this function:



17.49.1.2 calc_channel_elevation()

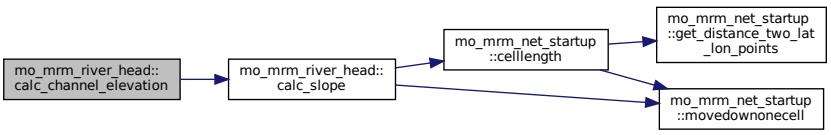
```
subroutine, public mo_mrm_river_head::calc_channel_elevation
```

Definition at line 67 of file mo_mrm_river_head.f90.

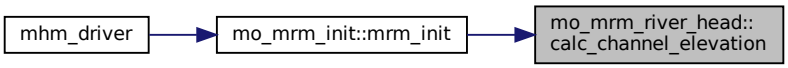
References calc_slope(), mo_common_variables::domainmeta, mo_mrm_global_variables::l0_channel_depth, mo_common_variables::l0_elev, mo_mrm_global_variables::l0_facc, mo_mrm_global_variables::l0_fdir, and mo_common_constants::nodata_i4.

Referenced by mo_mrm_init::mrm_init().

Here is the call graph for this function:



Here is the caller graph for this function:



17.49.1.3 calc_river_head()

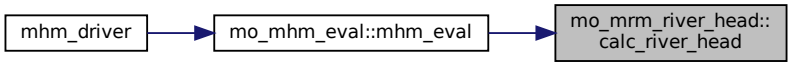
```
subroutine, public mo_mrm_river_head::calc_river_head (
             integer(i4), intent(in) iDomain,
             real(dp), dimension(:), intent(in) L11_Qmod,
             real(dp), dimension(:), intent(inout), allocatable river_head )
```

Definition at line 133 of file mo_mrm_river_head.f90.

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().

Here is the caller graph for this function:



17.49.1.4 calc_slope()

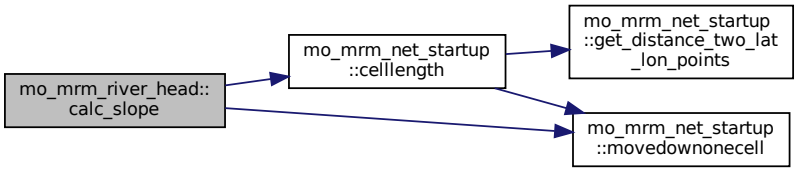
```
real(dp) function mo_mrm_river_head::calc_slope (
             integer(i4), intent(in) iDomain,
             real(dp), dimension(:,:), intent(in), allocatable elev0,
             integer(i4), dimension(:,:), intent(in), allocatable fDir0,
             integer(i4), intent(in) i,
             integer(i4), intent(in) j )  [private]
```

Definition at line 162 of file mo_mrm_river_head.f90.

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:

17.49.1.5 create_output()

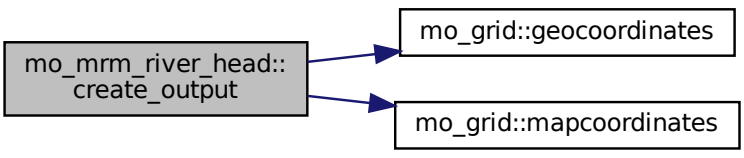
```
subroutine, public mo_mrm_river_head::create_output (
             integer(i4), intent(in) iDomain,
             character(256), intent(in) OutPath )
```

Definition at line 203 of file mo_mrm_river_head.f90.

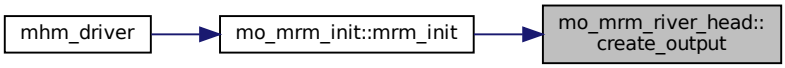
References mo_common_variables::contact, mo_common_variables::conventions, mo_common_mhm_mrm_↩ variables::evalper, mo_grid::geocoordinates(), mo_common_variables::history, mo_grid::mapcoordinates(), mo↩ _common_variables::mhm_details, nc_riverhead, nc_time, mo_common_constants::nodata_dp, mo_common↩ _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:



17.49.1.6 init_masked_zeros_l0()

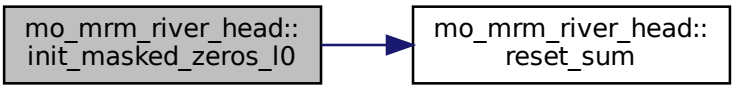
```
subroutine, public mo_mrm_river_head::init_masked_zeros_l0 (
             integer(i4), intent(in) iDomain,
             real(dp), dimension(:), intent(inout), allocatable data )
```

Definition at line 38 of file mo_mrm_river_head.f90.

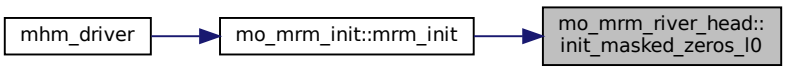
References mo_common_variables::level0, mo_common_constants::nodata_dp, and reset_sum().

Referenced by mo_mrm_init::mrm_init().

Here is the call graph for this function:



Here is the caller graph for this function:



17.49.1.7 reset_sum()

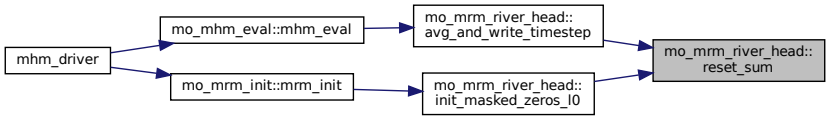
```
subroutine, private mo_mrm_river_head::reset_sum (
             integer(i4), intent(in) iDomain,
             real(dp), dimension(:), intent(inout) data )  [private]
```

Definition at line 51 of file mo_mrm_river_head.f90.

References mo_common_variables::level0.

Referenced by avg_and_write_timestep(), and init_masked_zeros_l0().

Here is the caller graph for this function:



17.49.2 Variable Documentation

17.49.2.1 nc_riverhead

```
type(ncvariable), dimension(:), allocatable mo_mrm_river_head::nc_riverhead  [private]
```

Definition at line 28 of file mo_mrm_river_head.f90.

Referenced by avg_and_write_timestep(), and create_output().

17.49.2.2 nc_time

```
type(ncvariable), dimension(:), allocatable mo_mrm_river_head::nc_time  [private]
```

Definition at line 28 of file mo_mrm_river_head.f90.

Referenced by avg_and_write_timestep(), and create_output().

17.49.2.3 sum_counter

```
integer(i4), dimension(:), allocatable mo_mrm_river_head::sum_counter  [private]
```

Definition at line 32 of file mo_mrm_river_head.f90.

Referenced by avg_and_write_timestep(), calc_river_head(), and create_output().

17.49.2.4 time_counter

```
integer(i4), dimension(:), allocatable mo_mrm_river_head::time_counter  [private]
```

Definition at line 30 of file mo_mrm_river_head.f90.

Referenced by avg_and_write_timestep(), and create_output().

17.50 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_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).

17.50.1 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

17.50.2 Function/Subroutine Documentation

17.50.2.1 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 domain (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 domain) 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

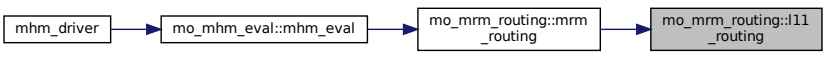
Date

Dec 2005

Definition at line 379 of file mo_mrm_routing.f90.

Referenced by mrm_routing().

Here is the caller graph for this function:



17.50.2.2 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

Parameters

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
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

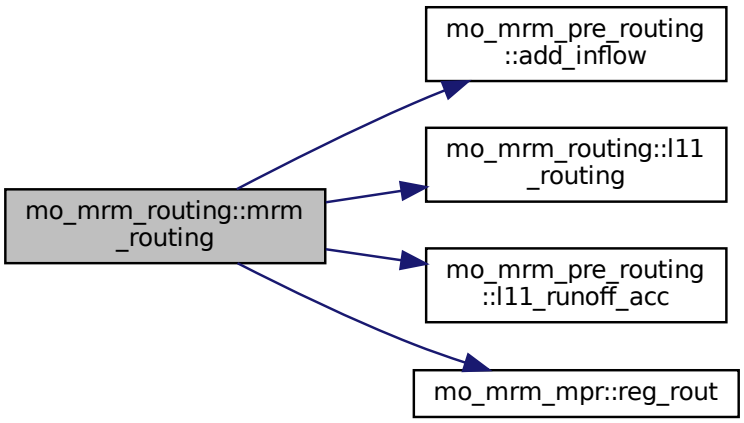
Aug 2015

Definition at line 103 of file mo_mrm_routing.f90.

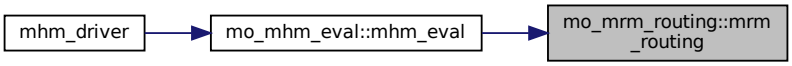
References mo_mrm_pre_routing::add_inflow(), mo_mrm_global_variables::is_start, l11_routing(), mo_mrm_pre↩_routing::l11_runoff_acc(), mo_mrm_mpr::reg_rout(), and mo_mrm_global_variables::riv_temp_pcs.

Referenced by mo_mhm_eval::mhm_eval().

Here is the call graph for this function:



Here is the caller graph for this function:



17.51 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, domain_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.

17.51.1 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

17.51.2 Function/Subroutine Documentation

17.51.2.1 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

Parameters

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

Definition at line 71 of file mo_mrm_signatures.f90.

17.51.2.2 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:10.1002/2014↩ WR016520

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)
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

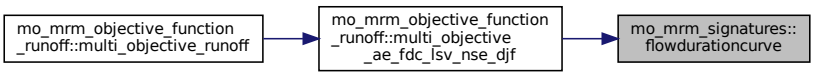
Date

March 2014

Definition at line 201 of file mo_mrm_signatures.f90.

Referenced by mo_mrm_objective_function_runoff::multi_objective_ae_fdc_lsv_nse_djf().

Here is the caller graph for this function:



17.51.2.3 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

Definition at line 341 of file mo_mrm_signatures.f90.

17.51.2.4 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) $\f 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

Definition at line 513 of file mo_mrm_signatures.f90.

17.51.2.5 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

Definition at line 660 of file mo_mrm_signatures.f90.

17.51.2.6 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

Definition at line 788 of file mo_mrm_signatures.f90.

17.51.2.7 runoffratio()

```
real(dp) function, public mo_mrm_signatures::runoffratio (
            real(dp), dimension(:), intent(in) data,
            real(dp), intent(in) domain_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 domain 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) :: domain_area*	area of domain [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

Definition at line 925 of file mo_mrm_signatures.f90.

17.51.2.8 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

Definition at line 1045 of file mo_mrm_signatures.f90.

17.52 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 (iDomain, nCells, timeStep_model_outputs, domainDateTime, tt, 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

- type(outputdataset) nc

17.52.1 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

17.52.2 Function/Subroutine Documentation

17.52.2.1 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

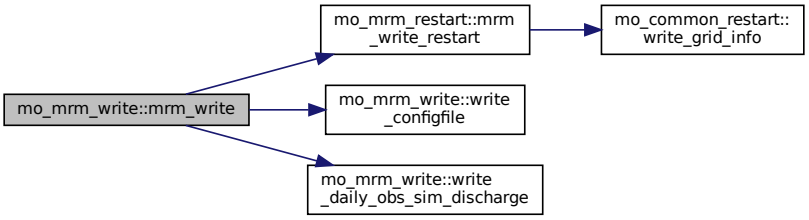
Aug 2015

Definition at line 54 of file mo_mrm_write.f90.

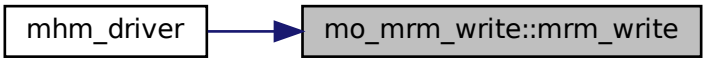
References mo_mrm_global_variables::domain_mrm, mo_common_variables::domainmeta, mo_common_mhm↩_mrm_variables::evalper, mo_mrm_global_variables::gauge, mo_common_mhm_mrm_variables::mrm_coupling↩_mode, mo_mrm_global_variables::mrm_runoff, mo_mrm_restart::mrm_write_restart(), mo_common_variables↩::mrmfilerestartout, 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().

Here is the call graph for this function:



Here is the caller graph for this function:



17.52.2.2 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

Definition at line 753 of file mo_mrm_write.f90.

References mo_common_variables::dirconfigout, mo_common_mhm_mrm_file::file_opti, and mo_common_↩ mhm_mrm_file::uopti.

17.52.2.3 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

Definition at line 832 of file mo_mrm_write.f90.

References mo_common_variables::dirconfigout, mo_common_mhm_mrm_file::file_opti_nml, mo_common_↩ variables::processmatrix, and mo_common_mhm_mrm_file::uopti_nml.

17.52.2.4 mrm_write_output_fluxes()

```
subroutine, public mo_mrm_write::mrm_write_output_fluxes (
             integer(i4), intent(in) iDomain,
             integer(i4), intent(in) nCells,
             integer(i4), intent(in) timeStep_model_outputs,
             type(datetimeinfo), intent(in) domainDateTime,
             integer(i4), intent(in) tt,
             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) :: iDomain*	
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

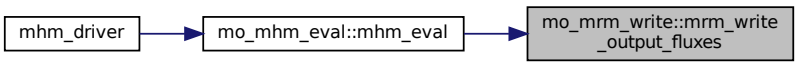
Aug 2015

Definition at line 666 of file mo_mrm_write.f90.

References nc, and mo_mrm_global_variables::riv_temp_pcs.

Referenced by mo_mhm_eval::mhm_eval().

Here is the caller graph for this function:



17.52.2.5 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

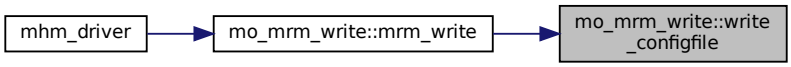
May 2013

Definition at line 156 of file mo_mrm_write.f90.

References mo_common_variables::dirconfigout, mo_mrm_global_variables::dirgauges, mo_common_variables::dirlcover, mo_common_variables::dirmorpho, mo_common_variables::dirout, mo_mrm_global_variables::dirtotalrunoff, mo_mrm_global_variables::domain_mrm, mo_common_variables::domainmeta, 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_mrm_global_variables::inflowgauge, 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_common_mhm_mrm_variables::mrm_coupling_mode, mo_common_variables::mrmfilerestartout, 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 caller graph for this function:



17.52.2.6 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)
in	*real(dp), dimension(:, :) :: Qsim*	daily time series of modeled dischargedims = (nModeling_days , nGauges_total)

Authors

Rohini Kumar

Date

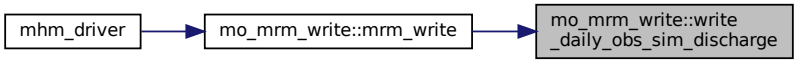
August 2013

Definition at line 476 of file mo_mrm_write.f90.

References mo_common_variables::dirout, mo_mrm_global_variables::domain_mrm, mo_common_variables::domainmeta, mo_common_mhm_mrm_variables::evalper, mo_mrm_file::file_daily_discharge, mo_mrm_global_variables::gauge, mo_mrm_file::ncfile_discharge, mo_common_constants::nodata_dp, and mo_mrm_file::udaily_discharge.

Referenced by mrm_write().

Here is the caller graph for this function:



17.52.3 Variable Documentation

17.52.3.1 nc

```
type(outputdataset) mo_mrm_write::nc  [private]
```

Definition at line 29 of file mo_mrm_write.f90.

Referenced by mrm_write_output_fluxes().

17.53 mo_mrm_write_fluxes_states Module Reference

Creates NetCDF output for different fluxes and state variables of mHM.

Data Types

- interface outputvariable
- interface outputdataset

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 (iDomain, mask, nCells)

  Initialize OutputDataset.
- subroutine updatedataset (self, sidx, eidx, L11_Qmod, L11_riv_temp)

  Update all variables.
- subroutine writetimestep (self, timestep)

  Write all accumulated data.
- subroutine close (self)

  Close the file.
- type(ncdataset) function createoutputfile (iDomain)

  Create and initialize output file for X & Y coordinate system.
- subroutine writevariableattributes (var, long_name, unit)

  Write output variable attributes.

17.53.1 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

17.53.2 Function/Subroutine Documentation

17.53.2.1 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

Definition at line 452 of file mo_mrm_write_fluxes_states.f90.

References mo_common_variables::dirout.

17.53.2.2 createoutputfile()

```
type(ncdataset) function mo_mrm_write_fluxes_states::createoutputfile (
             integer(i4), intent(in) iDomain )
```

Create and initialize output file for X & Y coordinate system.

Create output file, write all non-dynamic variables and global attributes for the given domain for X & Y coordinate system

Returns

type(NcDataset)

Parameters

in	*integer(i4) :: iDomain*	-> domain id

Authors

David Schaefer

Date

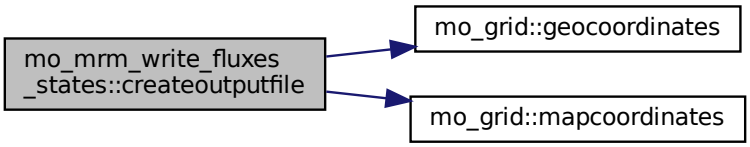
June 2015

Definition at line 494 of file mo_mrm_write_fluxes_states.f90.

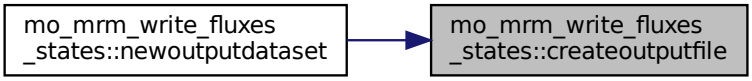
References mo_common_variables::dirout, mo_common_mhm_mrm_variables::evalper, mo_mrm_file::file_mrm_output, mo_grid::geocoordinates(), mo_common_variables::iflag_cordinate_sys, mo_mrm_global_variables::level11, mo_grid::mapcoordinates(), mo_common_constants::nodata_dp, and mo_mrm_file::version.

Referenced by newoutputdataset().

Here is the call graph for this function:



Here is the caller graph for this function:



17.53.2.3 newoutputdataset()

```
type(outputdataset) function mo_mrm_write_fluxes_states::newoutputdataset (
            integer(i4), intent(in) iDomain,
            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) :: iDomain*	-> domain id
in	*logical, dimension(:, :) :: mask*	
in	*integer(i4) :: nCells*	

Authors

Matthias Zink

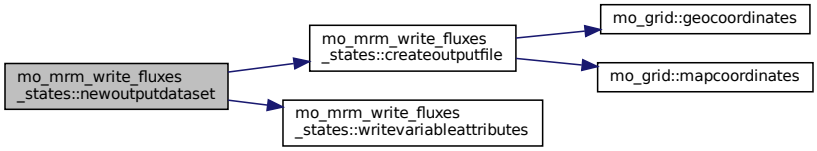
Date

Apr 2013

Definition at line 240 of file mo_mrm_write_fluxes_states.f90.

References createoutputfile(), mo_common_variables::iflag_cordinate_sys, mo_mrm_global_variables::outputflxstate↩
_mrm, mo_mrm_global_variables::riv_temp_pcs, and writevariableattributes().

Here is the call graph for this function:



17.53.2.4 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*	
in	*character(16), dimension(3) :: dims*	
in	*integer(i4) :: ncells*	-> number of cells in domain
in	*logical, dimension(:, :) :: mask*	
in	*logical, optional :: avg*	-> average the data before writing

Authors

David Schaefer

Date

June 2015

Definition at line 100 of file mo_mrm_write_fluxes_states.f90.

References mo_mrm_global_variables::output_deflate_level_mrm.

17.53.2.5 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,
             real(dp), dimension(:), intent(in), optional L11_riv_temp )
```

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 domain related data in L1_* arguments
in	*integer(i4) :: sidx, eidx*	- end index of the domain related data in L1_* arguments
in	*real(dp), dimension(:) :: L11_Qmod*	
in	*real(dp), dimension(:), optional :: L11_riv_temp*	

Authors

Matthias Zink

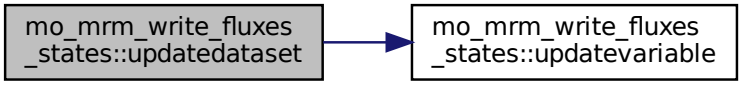
Date

Apr 2013

Definition at line 341 of file mo_mrm_write_fluxes_states.f90.

References mo_mrm_global_variables::outputflxstate_mrm, and updatevariable().

Here is the call graph for this function:



17.53.2.6 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

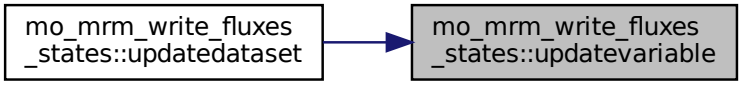
Date

June 2015

Definition at line 156 of file mo_mrm_write_fluxes_states.f90.

Referenced by updatedataset().

Here is the caller graph for this function:



17.53.2.7 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

Definition at line 408 of file mo_mrm_write_fluxes_states.f90.

17.53.2.8 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

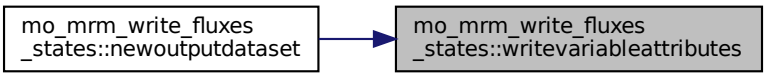
June 2015

Definition at line 649 of file mo_mrm_write_fluxes_states.f90.

References mo_common_constants::nodata_dp.

Referenced by newoutputdataset().

Here is the caller graph for this function:



17.53.2.9 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

Definition at line 193 of file mo_mrm_write_fluxes_states.f90.

References mo_common_constants::nodata_dp.

17.54 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.

17.54.1 Detailed Description

Multiscale parameter regionalization (MPR).

This module provides the routines for multiscale parameter regionalization (MPR).

Authors

Stephan Thober, Rohini Kumar

Date

Dec 2012

17.54.2 Function/Subroutine Documentation

17.54.2.1 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

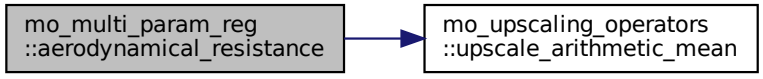
Apr 2013

Definition at line 1232 of file mo_multi_param_reg.f90.

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:

17.54.2.2 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

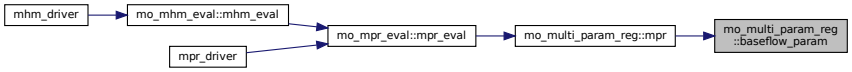
Dec 2012

Definition at line 716 of file mo_multi_param_reg.f90.

References mo_mpr_global_variables::geounitlist, and mo_common_constants::nodata_dp.

Referenced by mpr().

Here is the caller graph for this function:



17.54.2.3 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

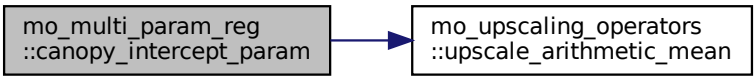
Aug. 2013

Definition at line 1103 of file mo_multi_param_reg.f90.

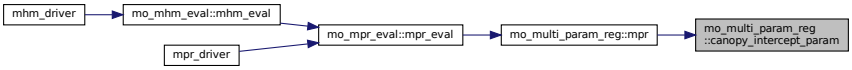
References mo_upscaling_operators::upscale_arithmetic_mean().

Referenced by mpr().

Here is the call graph for this function:



Here is the caller graph for this function:



17.54.2.4 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

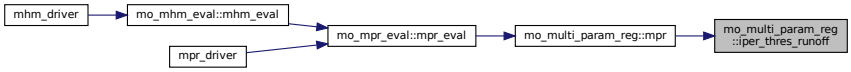
Date

Dec 2012

Definition at line 910 of file mo_multi_param_reg.f90.

Referenced by mpr().

Here is the caller graph for this function:



17.54.2.5 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
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

Parameters

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

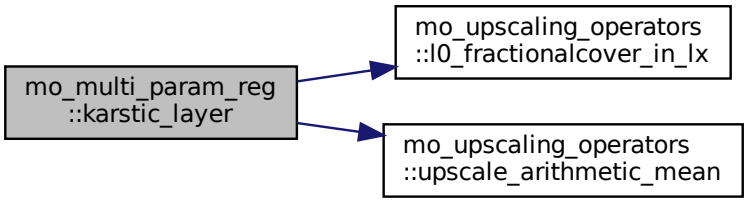
Feb 2013

Definition at line 971 of file mo_multi_param_reg.f90.

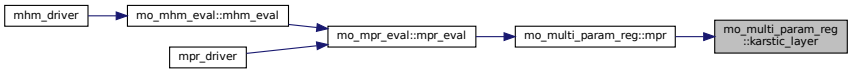
References mo_mpr_global_variables::geounitkar, mo_mpr_global_variables::geounitlist, mo_upscaling_↩ operators::l0_fractionalcover_in_lx(), mo_common_constants::nodata_dp, mo_common_constants::nodata_i4, and mo_upscaling_operators::upscale_arithmetic_mean().

Referenced by mpr().

Here is the call graph for this function:



Here is the caller graph for this function:



17.54.2.6 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

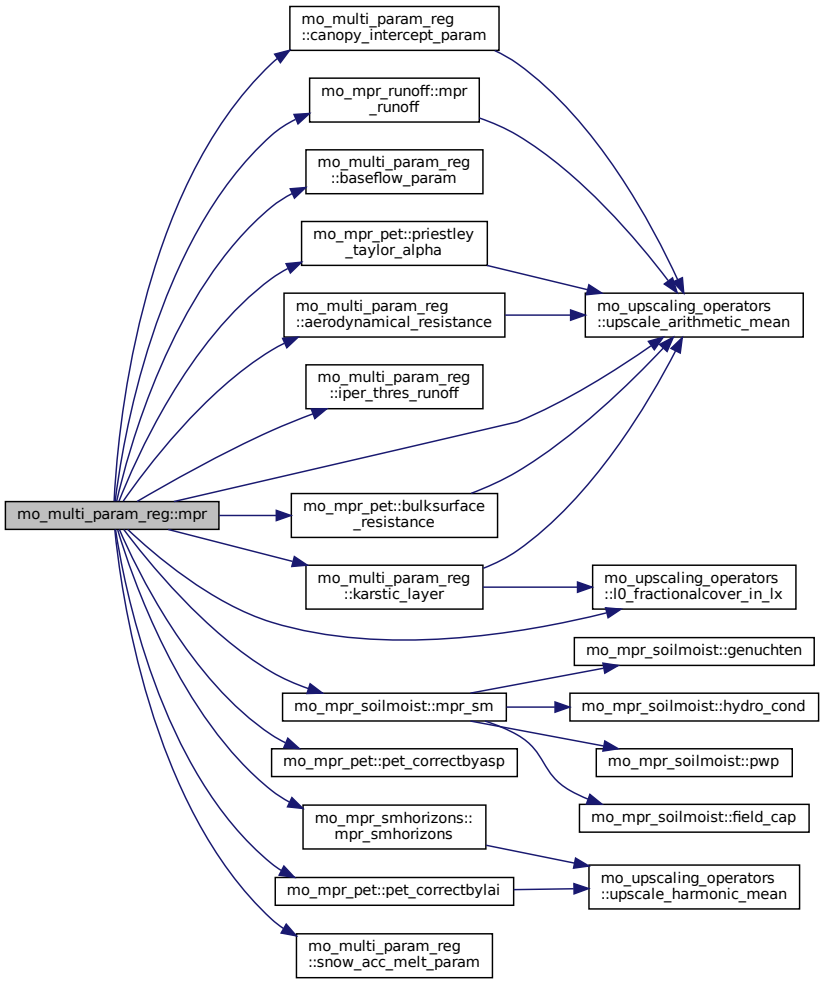
Dec 2012

Definition at line 124 of file mo_multi_param_reg.f90.

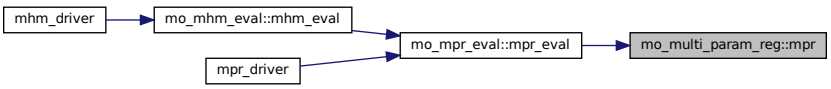
References aerodynamical_resistance(), baseflow_param(), mo_mpr_pet::bulksurface_resistance(), canopy_↩ intercept_param(), mo_mpr_global_variables::fracsealed_cityarea, mo_common_variables::global_parameters, mo_mpr_global_variables::horizondepth_mhm, mo_mpr_global_variables::iflag_soildb, iper_thres_runoff(), karstic_layer(), mo_upscaling_operators::l0_fractionalcover_in_lx(), mo_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:



17.54.2.7 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(:) :: fImperm1*	[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

Definition at line 827 of file mo_multi_param_reg.f90.

Referenced by mpr().

Here is the caller graph for this function:

17.55 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.

17.55.1 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.

17.55.2 Function/Subroutine Documentation

17.55.2.1 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*	
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

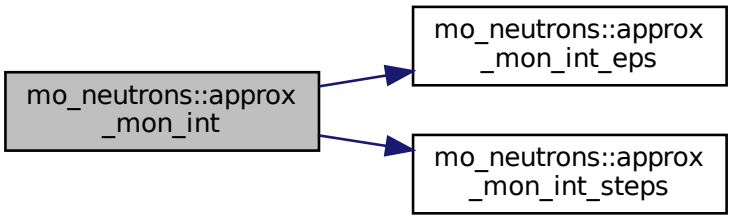
Jun 2018

Definition at line 436 of file mo_neutrons.f90.

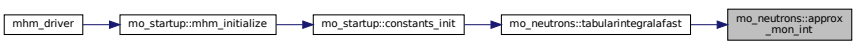
References approx_mon_int_eps(), and approx_mon_int_steps().

Referenced by tabularintegralafast().

Here is the call graph for this function:



Here is the caller graph for this function:



17.55.2.2 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

Definition at line 596 of file mo_neutrons.f90.

Referenced by approx_mon_int().

Here is the caller graph for this function:

17.55.2.3 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*	
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

Definition at line 526 of file mo_neutrons.f90.

Referenced by approx_mon_int().

Here is the caller graph for this function:

17.55.2.4 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

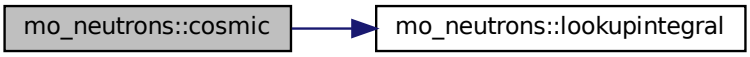
Mar 2015

Definition at line 145 of file mo_neutrons.f90.

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, and lookupintegral().

Referenced by mo_mhm::mhm().

Here is the call graph for this function:



Here is the caller graph for this function:

17.55.2.5 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:10.1029/2009↩ WR008726.

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

Definition at line 78 of file mo_neutrons.f90.

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:

17.55.2.6 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

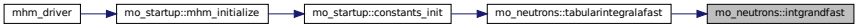
Date

Jun 2018

Definition at line 718 of file mo_neutrons.f90.

Referenced by tabularintegralafast().

Here is the caller graph for this function:



17.55.2.7 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

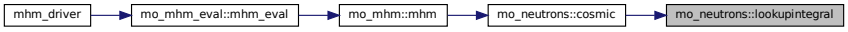
Date

Jun 2018

Definition at line 662 of file mo_neutrons.f90.

Referenced by cosmic().

Here is the caller graph for this function:



17.55.2.8 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

Definition at line 287 of file mo_neutrons.f90.

17.55.2.9 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

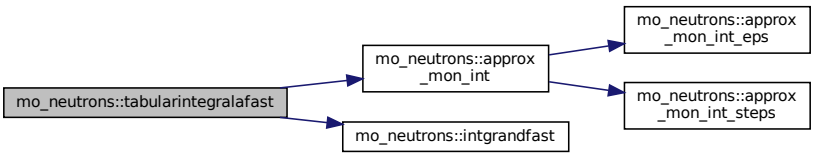
Nov 2017

Definition at line 368 of file mo_neutrons.f90.

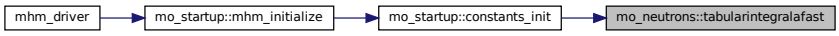
References approx_mon_int(), and intgrandfast().

Referenced by mo_startup::constants_init().

Here is the call graph for this function:



Here is the caller graph for this function:



17.56 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, public objective_master (parameterset, eval, arg1, arg2, arg3)

  Wrapper for objective functions.
- subroutine, public objective_subprocess (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, dimension(6) objective_q_et_tws_kge_catchment_avg (parameterset, eval)

  Objective function for et, tws and q.
- subroutine init_indexarray_for_opti_data (domainMeta, optidataOption, nOptiDomains, opti_domain_indices)

  creates an index array of the inidices of the domains eval should MPI process.
- 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 domain_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 domain_avg ET (standarized scores)
- subroutine create_domain_avg_tws (iDomain, twsOptiSim, tws_catch_avg_domain, tws_opti_catch_avg_↩ domain, mask_times)
- subroutine create_domain_avg_et (iDomain, etOptiSim, et_catch_avg_domain, et_opti_catch_avg_domain, mask_times)
- subroutine convert_tws_to_twsa (twsOptiSim, L1_twsaObs, twsaOptiSim)

17.56.1 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(domain_avg_TWS) - objective function using Q and domain average !ToDo: change comment (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

17.56.2 Function/Subroutine Documentation

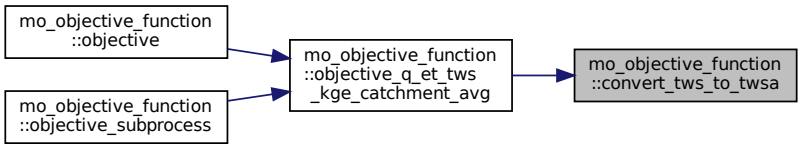
17.56.2.1 convert_tws_to_twsa()

```
subroutine mo_objective_function::convert_tws_to_twsa (
             type(optidata_sim), intent(in) twsOptiSim,
             type(optidata), intent(in) L1_twsaObs,
             type(optidata_sim), intent(inout) twsaOptiSim )
```

Definition at line 2601 of file mo_objective_function.f90.

Referenced by objective_q_et_tws_kge_catchment_avg().

Here is the caller graph for this function:



17.56.2.2 create_domain_avg_et()

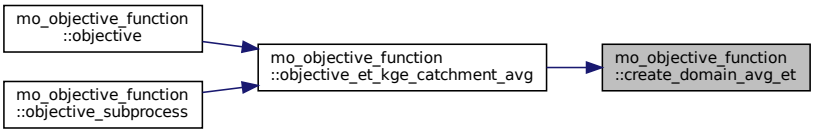
```
subroutine mo_objective_function::create_domain_avg_et (
             integer(i4), intent(in) iDomain,
             type(optidata_sim), dimension(:), intent(in) etOptiSim,
             real(dp), dimension(:), intent(out), allocatable et_catch_avg_domain,
             real(dp), dimension(:), intent(out), allocatable et_opti_catch_avg_domain,
             logical, dimension(:), intent(out), allocatable mask_times )
```

Definition at line 2546 of file mo_objective_function.f90.

References mo_global_variables::l1_etobs, and mo_common_constants::nodata_dp.

Referenced by objective_et_kge_catchment_avg().

Here is the caller graph for this function:



17.56.2.3 create_domain_avg_tws()

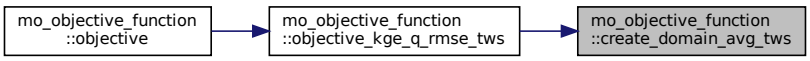
```
subroutine mo_objective_function::create_domain_avg_tws (
             integer(i4), intent(in) iDomain,
             type(optidata_sim), dimension(:), intent(in) twsOptiSim,
             real(dp), dimension(:), intent(out), allocatable tws_catch_avg_domain,
             real(dp), dimension(:), intent(out), allocatable tws_opti_catch_avg_domain,
             logical, dimension(:), intent(out), allocatable mask_times )
```

Definition at line 2491 of file mo_objective_function.f90.

References mo_global_variables::l1_twsaobs, and mo_common_constants::nodata_dp.

Referenced by objective_kge_q_rmse_tws().

Here is the caller graph for this function:



17.56.2.4 init_indexarray_for_opti_data()

```
subroutine mo_objective_function::init_indexarray_for_opti_data (
             type(domain_meta), intent(in) domainMeta,
             integer(i4), intent(in) optidataOption,
             integer(i4), intent(out) nOptiDomains,
             integer(i4), dimension(:), intent(out), allocatable opti_domain_indices )
```

creates an index array of the inidices of the domains eval should MPI process.

The data type domainMeta contains an array optidata of size domainMetanDomains, telling us, which domains should be optimized with which opti_data. This subroutine splits all domains assigned to a process and returns an index list corresponding to the value of domainMetaoptidata.

The index array opti_domain_indices can then be passed as an optional argument to the eval subroutine. The eval then instead of using loops over all domains only uses the passed indices.

This subroutine also returns the size of that array since it helps with the calculations of the optimization in the end.

Authors

Maren Kaluza

Date

July 2019

Parameters

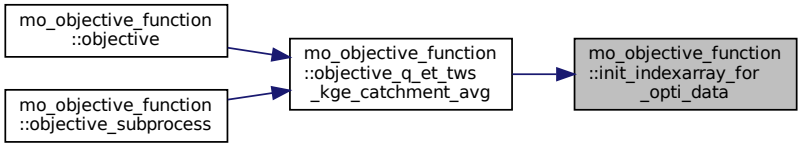
in	*domainmeta*	meta data for all domains assigned to that process
in	*optidataoption*	which opti data should be used in the eval called after calling this subroutine
out	*noptidomains*	number of domains that will be optimized in the following eval call
out	*opti_domain_indices*	the indices of the domains that are to be processed in the following eval call

domain loop counter

Definition at line 908 of file mo_objective_function.f90.

Referenced by objective_q_et_tws_kge_catchment_avg().

Here is the caller graph for this function:



17.56.2.5 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

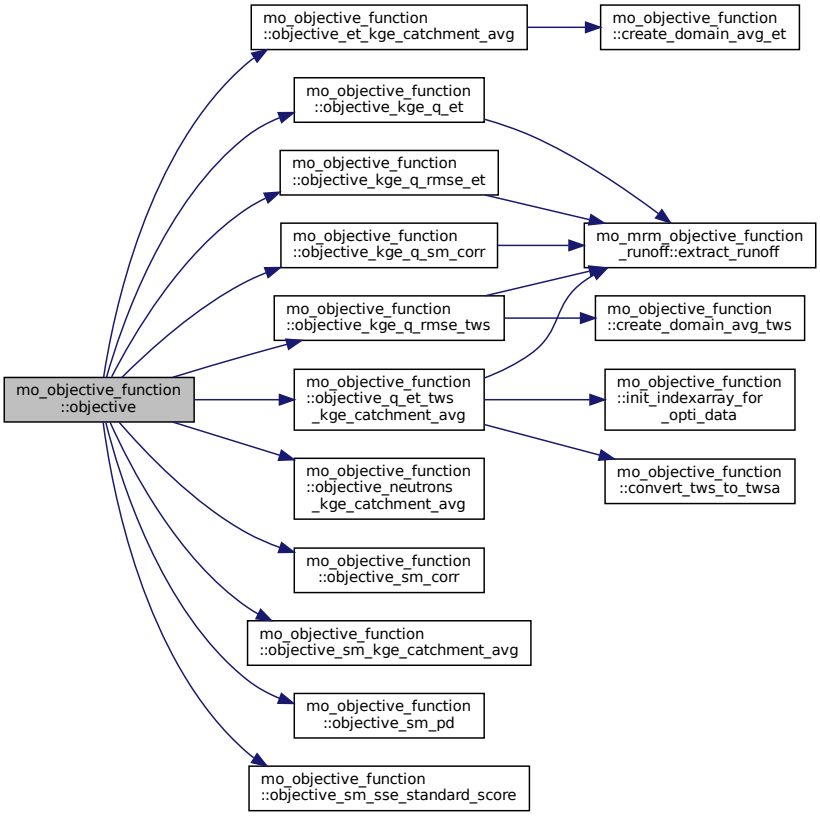
Date

Dec 2012

Definition at line 88 of file mo_objective_function.f90.

References 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_q_et_tws_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.

Here is the call graph for this function:



17.56.2.6 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 domain 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 domains 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

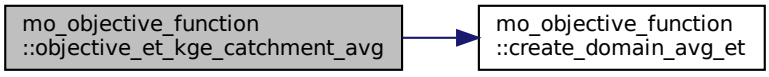
simulated et

Definition at line 1780 of file mo_objective_function.f90.

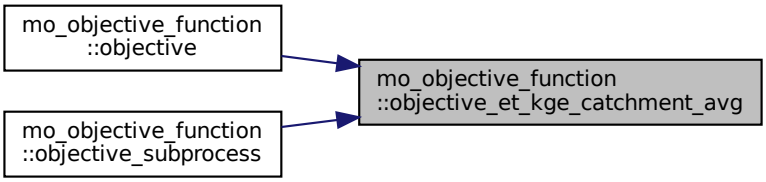
References create_domain_avg_et(), mo_common_variables::domainmeta, and mo_common_constants::nodata↩_dp.

Referenced by objective(), and objective_subprocess().

Here is the call graph for this function:



Here is the caller graph for this function:



17.56.2.7 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

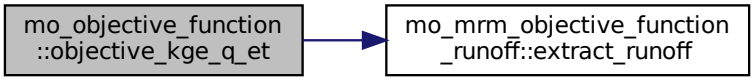
simulated et

Definition at line 2062 of file mo_objective_function.f90.

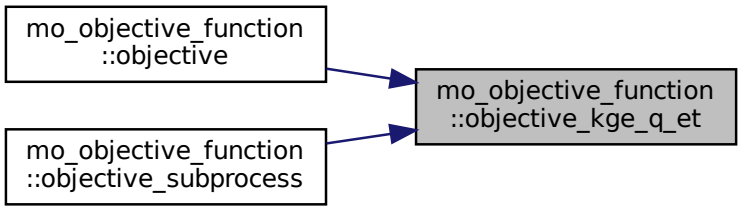
References mo_common_variables::domainmeta, mo_mrm_objective_function_runoff::extract_runoff(), mo_↩ global_variables::l1_etobs, and mo_common_variables::level1.

Referenced by objective(), and objective_subprocess().

Here is the call graph for this function:



Here is the caller graph for this function:



17.56.2.8 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 domain_avg ET (standarized scores)

Objective function of KGE for runoff and RMSE for domain_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

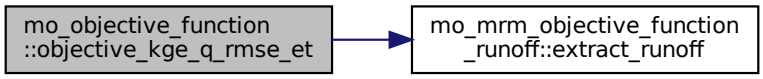
simulated et

Definition at line 2239 of file mo_objective_function.f90.

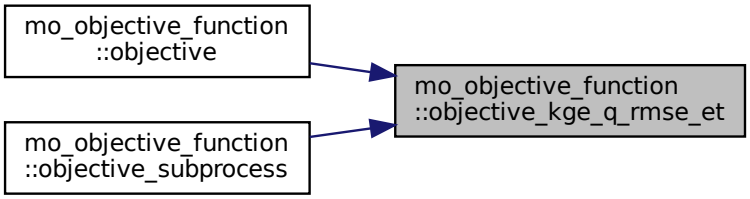
References mo_common_variables::domainmeta, mo_common_constants::eps_dp, mo_common_mhm_mrm_↩ variables::evalper, mo_mrm_objective_function_runoff::extract_runoff(), mo_global_variables::l1_etobs, and mo↩ _common_constants::nodata_dp.

Referenced by objective(), and objective_subprocess().

Here is the call graph for this function:



Here is the caller graph for this function:



17.56.2.9 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 domain_avg TWS (standarized scores)

Objective function of KGE for runoff and RMSE for domain_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

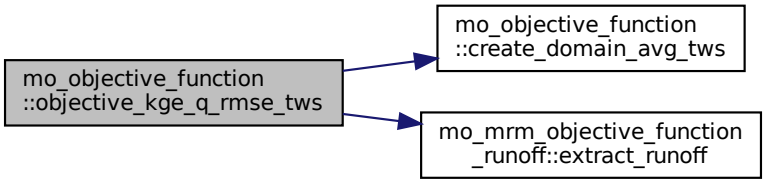
simulated tws

Definition at line 1396 of file mo_objective_function.f90.

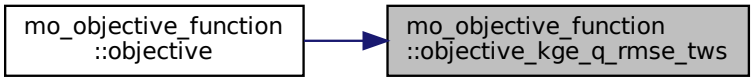
References create_domain_avg_tws(), mo_common_variables::domainmeta, mo_common_constants::eps_dp, mo_common_mhm_mrm_variables::evalper, mo_mrm_objective_function_runoff::extract_runoff(), mo_global_↩ variables::l1_twsaobs, and mo_common_constants::nodata_dp.

Referenced by objective().

Here is the call graph for this function:



Here is the caller graph for this function:



17.56.2.10 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

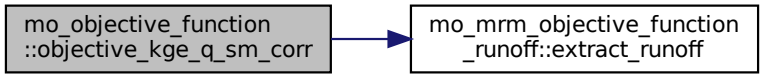
Mar. 2017

Definition at line 1882 of file mo_objective_function.f90.

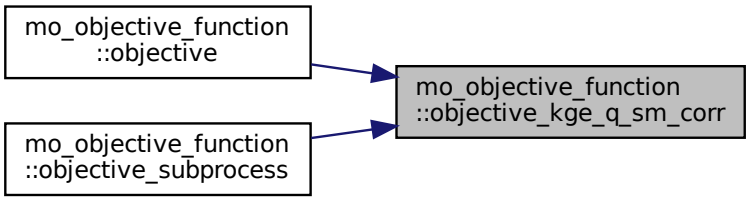
References mo_common_variables::domainmeta, mo_mrm_objective_function_runoff::extract_runoff(), mo_global_variables::l1_smobs, and mo_common_variables::level1.

Referenced by objective(), and objective_subprocess().

Here is the call graph for this function:



Here is the caller graph for this function:



17.56.2.11 objective_master()

```
real(dp) function, public mo_objective_function::objective_master (
             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 sends the parameterset to the subprocess, receives the partial objective and calculates the final objective

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

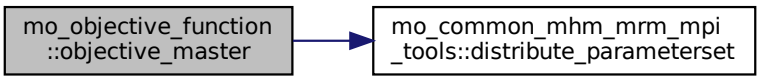
Date

Dec 2012

Definition at line 202 of file mo_objective_function.f90.

References mo_common_mhm_mrm_mpi_tools::distribute_parameterset(), mo_common_variables::domainmeta, mo_common_constants::nodata_dp, and mo_common_mhm_mrm_variables::opti_function.

Here is the call graph for this function:



17.56.2.12 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 domain 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 domains 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

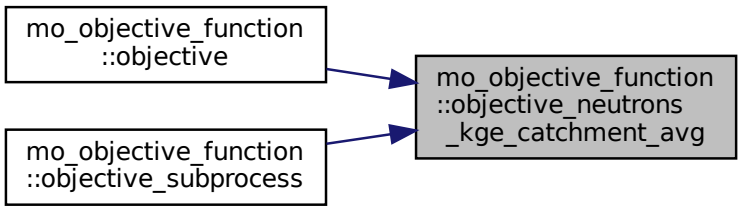
Jun 2015

Definition at line 1633 of file mo_objective_function.f90.

References mo_common_variables::domainmeta, mo_global_variables::l1_neutronsobs, and mo_common_↩ constants::nodata_dp.

Referenced by objective(), and objective_subprocess().

Here is the caller graph for this function:



17.56.2.13 objective_q_et_tws_kge_catchment_avg()

```
real(dp) function, dimension(6) mo_objective_function::objective_q_et_tws_kge_catchment_avg (
            real(dp), dimension(:), intent(in) parameterset,
            procedure(eval_interface), intent(in), pointer eval )
```

Objective function for et, tws and q.

The feature of this objective function is the separation of the eval call into four calls, each with another index list. The subroutine eval then only uses the indices from that index list internally instead of having loops over all domains. The integer array domainMetaoptidata decides which indices to use. Therefore the array is split into disjunct subsets, and, if chosen wisely in the namelist, also covers all domains.

With this the eval calls sum up in a way that for each domain eval was called at most once, but for different opti_data.

Authors

Maren Kaluza

Date

July 2019

Parameters

in	*parameterset*	the parameterset passed to the eval subroutine
in	*eval*	the eval subroutine called by this objective function

Returns

the return value of the objective function. In this case it is an array to provide the possibility to weight the outcome accordingly

domain loop counter

counter for short loops

for sixth root

modelled runoff for a given parameter set

number of all gauges, aquired via runoff

aggregated simulated runoff

measured runoff

mask for measured runoff

kge_q(nGaugesTotal)

gauges counter

number of q domains

number of et domains

number of tws domains

number of TWS and ET domains (providing both)

index array of ET domains

index array of TWS domains

index array of TWS and ET domains (providing both)

index array of ET domains

simulated et

simulated tws

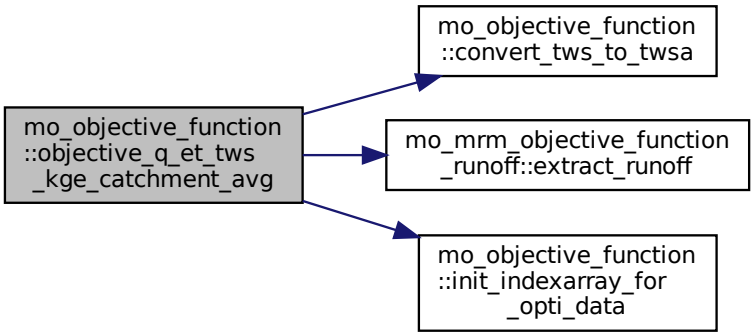
simulated twsa (anomaly)

Definition at line 649 of file mo_objective_function.f90.

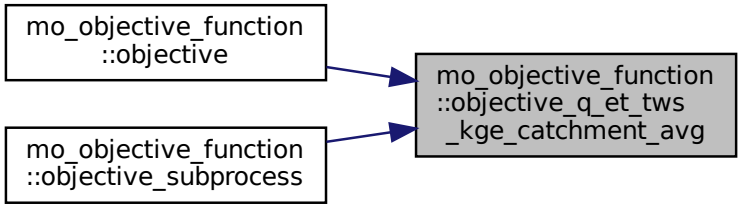
References convert_tws_to_twsa(), mo_common_variables::domainmeta, mo_mrm_objective_function_runoff↩ ::extract_runoff(), init_indexarray_for_opti_data(), mo_global_variables::l1_etobs, mo_global_variables::l1_↩ twsaobs, and mo_common_constants::nodata_dp.

Referenced by objective(), and objective_subprocess().

Here is the call graph for this function:



Here is the caller graph for this function:



17.56.2.14 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 domains 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

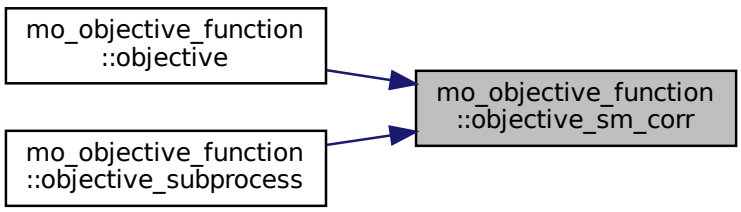
March 2015

Definition at line 986 of file mo_objective_function.f90.

References mo_common_variables::domainmeta, mo_global_variables::l1_smobs, and mo_common_variables::level1.

Referenced by objective(), and objective_subprocess().

Here is the caller graph for this function:



17.56.2.15 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 domain 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 domains 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

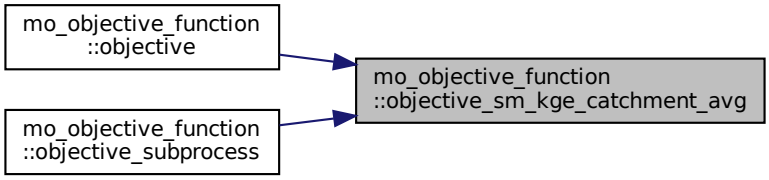
May 2015

Definition at line 504 of file mo_objective_function.f90.

References mo_common_variables::domainmeta, mo_global_variables::l1_smobs, mo_common_variables::level1, and mo_common_constants::nodata_dp.

Referenced by objective(), and objective_subprocess().

Here is the caller graph for this function:



17.56.2.16 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 domains 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

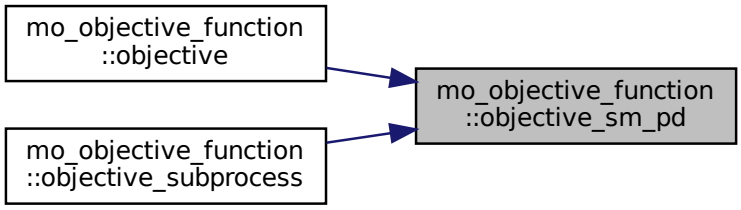
May 2015

Definition at line 1120 of file mo_objective_function.f90.

References mo_common_variables::domainmeta, mo_global_variables::l1_smobs, mo_common_variables::level1, and mo_common_constants::nodata_dp.

Referenced by objective(), and objective_subprocess().

Here is the caller graph for this function:



17.56.2.17 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 domains 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

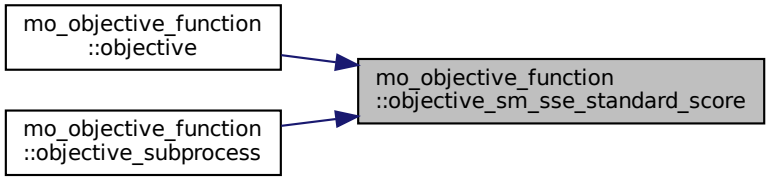
March 2015

Definition at line 1276 of file mo_objective_function.f90.

References mo_common_variables::domainmeta, mo_global_variables::l1_smobs, and mo_common_variables::level1.

Referenced by objective(), and objective_subprocess().

Here is the caller graph for this function:



17.56.2.18 objective_subprocess()

```
subroutine, public mo_objective_function::objective_subprocess (
             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 function receives the parameterset from the master process, selects the objective function case defined in a namelist, i.e. the global variable *opti_function*. It returns the partial objective function value for a specific parameter set.

Parameters

in	*REAL(dp), DIMENSION(:) :: parameterset*	

Parameters

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

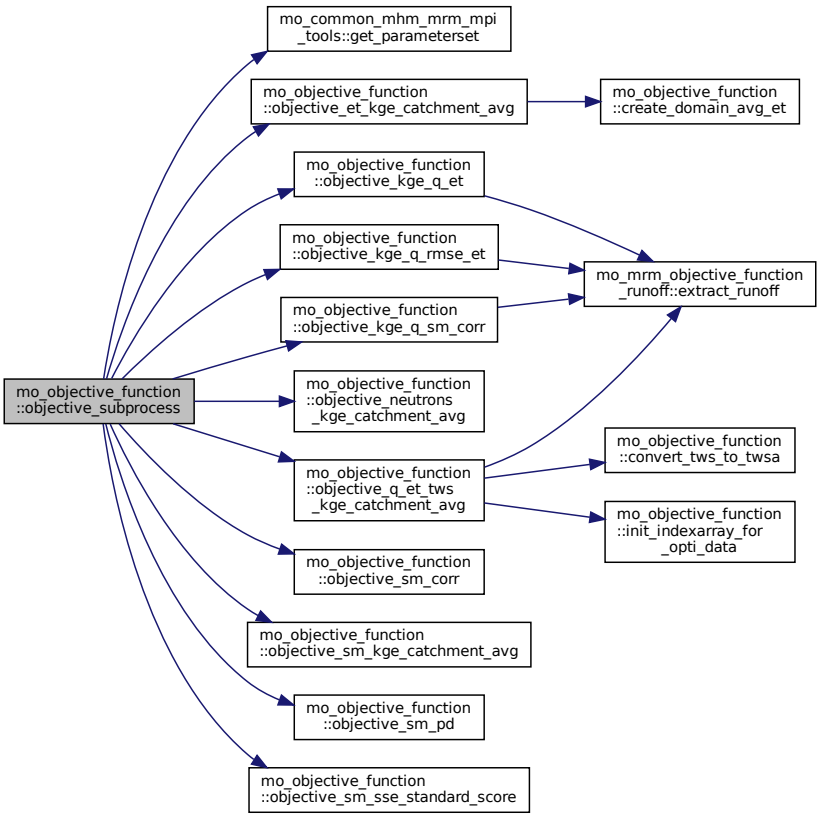
Date

Dec 2012

Definition at line 354 of file mo_objective_function.f90.

References mo_common_variables::domainmeta, mo_common_mhm_mrm_mpi_tools::get_parameterset(), mo↩_common_constants::nodata_dp, objective_et_kge_catchment_avg(), objective_kge_q_et(), objective_kge_q↩_rmse_et(), objective_kge_q_sm_corr(), objective_neutrons_kge_catchment_avg(), objective_q_et_tws_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.

Here is the call graph for this function:



17.57 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.

17.57.1 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

17.57.2 Function/Subroutine Documentation

17.57.2.1 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

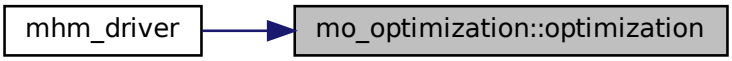
Oct 2015

Definition at line 51 of file mo_optimization.f90.

References mo_common_mhm_mrm_variables::dds_r, mo_common_variables::domainmeta, mo_common_variables::global_parameters, mo_common_mhm_mrm_variables::mcmc_error_params, mo_common_mhm_mrm_variables::mcmc_opti, 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_common_mhm_mrm_variables::sce_ngs, mo_common_mhm_mrm_variables::sce_npg, mo_common_mhm_mrm_variables::sce_nps, and mo_common_mhm_mrm_variables::seed.

Referenced by mhm_driver().

Here is the caller graph for this function:



17.58 mo_pet Module Reference

Module for calculating reference/potential evapotranspiration [mm d-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, public extraterr_rad_approx (doy, latitude)

  Approximation of extraterrestrial radiation.
- elemental pure real(dp) function, public slope_satpressure (tavg)

  slope of saturation vapour pressure curve
- elemental pure real(dp) function, public sat_vap_pressure (tavg)

  calculation of the saturation vapour pressure

17.58.1 Detailed Description

Module for calculating reference/potential evapotranspiration [mm d-1].

This module calculates PET [mm/d] 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

17.58.2 Function/Subroutine Documentation

17.58.2.1 extraterr_rad_approx()

```
elemental pure real(dp) function, public mo_pet::extraterr_rad_approx (
             integer(i4), intent(in) doy,
             real(dp), intent(in) latitude )
```

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

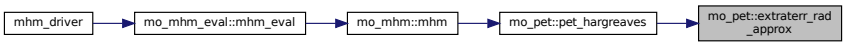
Apr 2014

Definition at line 315 of file mo_pet.f90.

References mo_mhm_constants::duffiedelta1, mo_mhm_constants::duffiedelta2, and mo_mhm_constants::duffiedr.

Referenced by pet_hargreaves().

Here is the caller graph for this function:



17.58.2.2 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 d-1]

Authors

Matthias Zink

Date

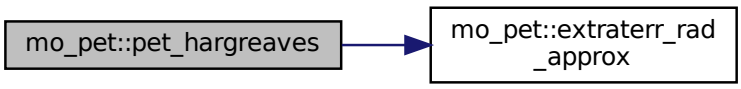
Dec 2012

Definition at line 74 of file mo_pet.f90.

References extraterr_rad_approx().

Referenced by mo_mhm::mhm().

Here is the call graph for this function:



Here is the caller graph for this function:

17.58.2.3 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 d-1]

Authors

Matthias Zink

Date

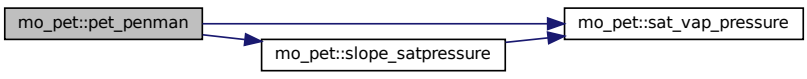
Apr 2014

Definition at line 236 of file mo_pet.f90.

References sat_vap_pressure(), and slope_satpressure().

Referenced by mo_mhm::mhm().

Here is the call graph for this function:



Here is the caller graph for this function:

17.58.2.4 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 d-1]

Authors

Matthias Zink

Date

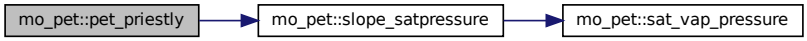
Apr 2014

Definition at line 150 of file mo_pet.f90.

References slope_satpressure().

Referenced by mo_mhm::mhm().

Here is the call graph for this function:



Here is the caller graph for this function:

17.58.2.5 sat_vap_pressure()

```
elemental pure real(dp) function, public mo_pet::sat_vap_pressure (
            real(dp), intent(in) tavg )
```

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

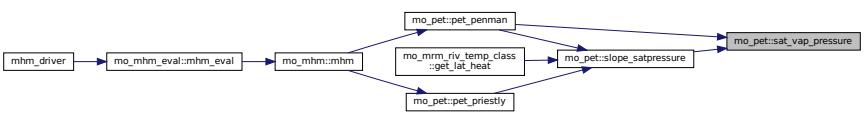
Apr 2014

Definition at line 427 of file mo_pet.f90.

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:



17.58.2.6 slope_satpressure()

```
elemental pure real(dp) function, public mo_pet::slope_satpressure (
            real(dp), intent(in) tavg )
```

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 $[^\circ C]$

Returns

real(dp) :: slope_satpressure — slope of saturation vapour pressure curve $[kPa\ K-1]$

Authors

Matthias Zink

Date

Apr 2014

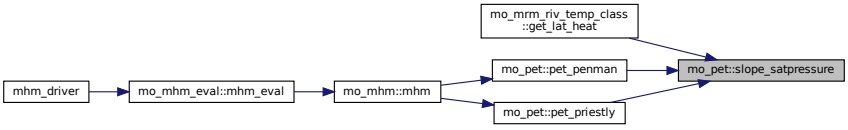
Definition at line 385 of file mo_pet.f90.

References sat_vap_pressure(), mo_mhm_constants::satpressureslope1, and mo_mhm_constants::tetens_c3.

Referenced by mo_mrm_riv_temp_class::get_lat_heat(), pet_penman(), and pet_priestly().

Here is the call graph for this function:

Here is the caller graph for this function:



17.59 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 (iDomain, nrows, ncols, mask, LAIPer_iDomain)

  Prepare gridded daily LAI data.
- subroutine, public prepare_gridded_mean_monthly_lai_data (iDomain, nrows, ncols, mask)

  prepare_gridded_mean_monthly_LAI_data

17.59.1 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

17.59.2 Function/Subroutine Documentation

17.59.2.1 prepare_gridded_daily_lai_data()

```
subroutine, public mo_prepare_gridded_lai::prepare_gridded_daily_lai_data (
             integer(i4), intent(in) iDomain,
             integer(i4), intent(in) nrows,
             integer(i4), intent(in) ncols,
             logical, dimension(:, :), intent(in) mask,
             type(period), intent(in), optional LAIPer_iDomain )
```

Prepare gridded daily LAI data.

Prepare gridded daily LAI data at Level-0 (e.g., using MODIS datasets)

Parameters

in	*integer(i4) :: iDomain, nrows, ncols*	domain Id
in	*integer(i4) :: iDomain, nrows, ncols*	domain Id
in	*integer(i4) :: iDomain, nrows, ncols*	domain Id
in	*logical, dimension(:, :) :: mask*	
in	*type(period), optional :: LAIPer_iDomain*	

Authors

John Craven & Rohini Kumar

Date

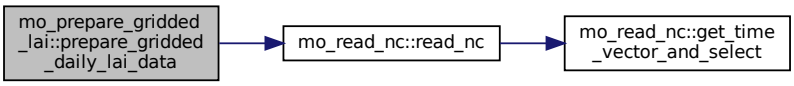
Aug 2013

Definition at line 61 of file mo_prepare_gridded_lai.f90.

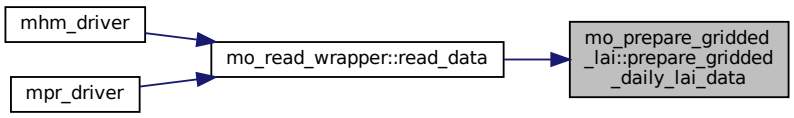
References mo_mpr_global_variables::dirgridded_lai, mo_mpr_global_variables::inputformat_gridded_lai, mo_mpr_global_variables::l0_gridded_lai, mo_mpr_global_variables::laiboundaries, mo_mpr_global_variables::nlai, mo_read_nc::read_nc(), and mo_mpr_global_variables::timestep_lai_input.

Referenced by mo_read_wrapper::read_data().

Here is the call graph for this function:



Here is the caller graph for this function:



17.59.2.2 prepare_gridded_mean_monthly_lai_data()

```
subroutine, public mo_prepare_gridded_lai::prepare_gridded_mean_monthly_lai_data (
            integer(i4), intent(in) iDomain,
            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) :: iDomain, nrows, ncols*	domain Id
in	*integer(i4) :: iDomain, nrows, ncols*	domain Id
in	*integer(i4) :: iDomain, nrows, ncols*	domain Id
in	*logical, dimension(:, :) :: mask*	

Authors

Rohini Kumar

Date

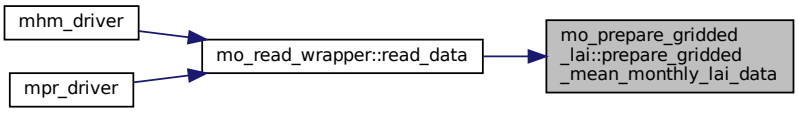
Dec 2016

Definition at line 151 of file mo_prepare_gridded_lai.f90.

References mo_mpr_global_variables::dirgridded_lai, mo_mpr_global_variables::l0_gridded_lai, mo_mpr_global_variables::laiboundaries, and mo_mpr_global_variables::nlai.

Referenced by mo_read_wrapper::read_data().

Here is the caller graph for this function:



17.60 mo_read_latlon Module Reference

reading latitude and longitude coordinates for each domain

Functions/Subroutines

- subroutine, public read_latlon (ii, lon_var_name, lat_var_name, level_name, level)

  reads latitude and longitude coordinates

17.60.1 Detailed Description

reading latitude and longitude coordinates for each domain

TODO: add description

Authors

Stephan Thober

Date

Nov 2013

17.60.2 Function/Subroutine Documentation

17.60.2.1 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 domain and appends it to the global variables latitude and longitude.

Parameters

in	*integer(i4) :: ii*	domain indexFile name of the domains must be xxx_latlon.nc, wherexxx is the domain 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

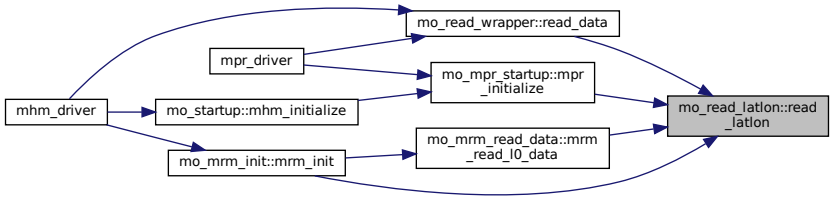
Nov 2013

Definition at line 65 of file mo_read_latlon.f90.

References mo_common_variables::filelatlon.

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 caller graph for this function:



17.61 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.

17.61.1 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

17.61.2 Function/Subroutine Documentation

17.61.2.1 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

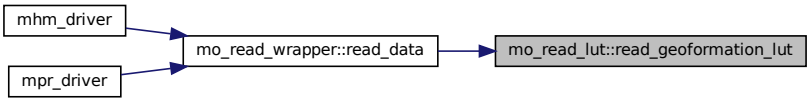
Date

Jan 2013

Definition at line 71 of file mo_read_lut.f90.

Referenced by mo_read_wrapper::read_data().

Here is the caller graph for this function:



17.61.2.2 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

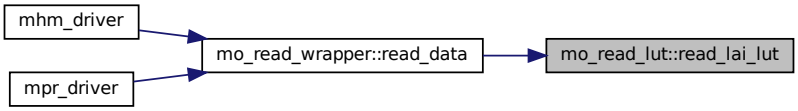
Date

Jan 2013

Definition at line 151 of file mo_read_lut.f90.

Referenced by mo_read_wrapper::read_data().

Here is the caller graph for this function:



17.62 mo_read_nc Module Reference

Reads forcing input data.

Functions/Subroutines

- subroutine, public read_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_nc (folder, nRows, nCols, varName, data, fileName)

  Reads time independent forcing input in NetCDF file format.
- 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.

17.62.1 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

17.62.2 Function/Subroutine Documentation

17.62.2.1 get_time_vector_and_select()

```
subroutine mo_read_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

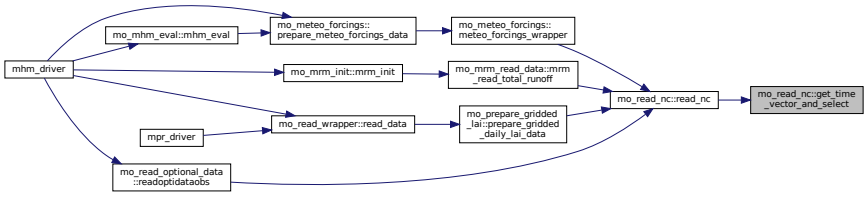
Date

Oct 2012

Definition at line 602 of file mo_read_nc.f90.

Referenced by read_nc().

Here is the caller graph for this function:



17.62.2.2 read_const_nc()

```
subroutine, public mo_read_nc::read_const_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,
character(256), intent(in), optional fileName )
```

Reads time independent forcing input in NetCDF file format.

Reads time independent 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 matrix dim_1 = longitude, dim_2 = latitude
in	*character(256), optional :: fileName*	name of file, defaults to varName data points

Note

Files have to be called like defined in mo_files. Furthermore the variable names have to be called like they are defined in the declaration of this subroutine. The NetCDF file has to have 2 dimensions:

1. x, 2. y, It is expected that the variables (especially)within the NetCDF files contain an unit attribute. The timestep has to be equidistant.

Author

Lennart Schueler, heavily influenced by read_nc

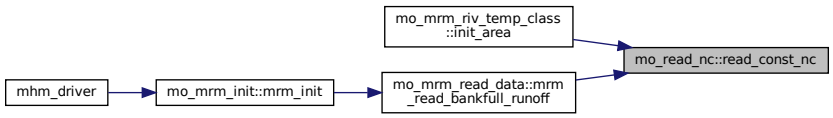
Date

May 2018

Definition at line 318 of file mo_read_nc.f90.

Referenced by mo_mrm_riv_temp_class::init_area(), and mo_mrm_read_data::mrm_read_bankfull_runoff().

Here is the caller graph for this function:



17.62.2.3 read_nc()

```
subroutine, public mo_read_nc::read_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 file, defaults to varName
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

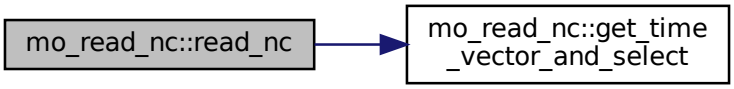
May 2013

Definition at line 85 of file mo_read_nc.f90.

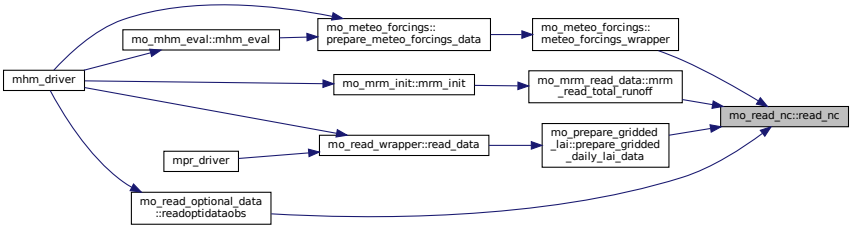
References get_time_vector_and_select().

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(), and mo_read_optional_data::readoptidataobs().

Here is the call graph for this function:



Here is the caller graph for this function:



17.62.2.4 read_weights_nc()

```
subroutine, public mo_read_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

Parameters

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
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

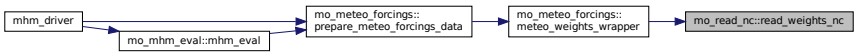
Date

Jan 2017

Definition at line 422 of file mo_read_nc.f90.

Referenced by mo_meteo_forcings::meteo_weights_wrapper().

Here is the caller graph for this function:



17.63 mo_read_optional_data Module Reference

Read optional data for mHM calibration.

Functions/Subroutines

- subroutine, public readoptidataobs (iDomain, domainID, L1_optiObs)

  Read evapotranspiration data from NetCDF file for calibration.

17.63.1 Detailed Description

Read optional data for mHM calibration.

Data have to be provided in resolution of the hydrology.

Authors

Matthias Zink

Date

Mar 2015

17.63.2 Function/Subroutine Documentation

17.63.2.1 readoptidataobs()

```
subroutine, public mo_read_optional_data::readoptidataobs (
             integer(i4), intent(in) iDomain,
             integer(i4), intent(in) domainID,
             type(optidata), intent(inout) L1_optiObs )
```

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) :: iDomain*	domain Id

Authors

Johannes Brenner

Date

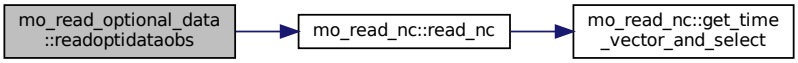
Feb 2017

Definition at line 54 of file mo_read_optional_data.f90.

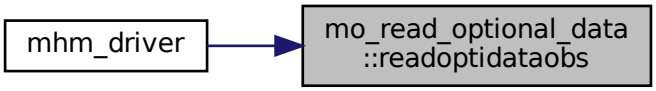
References mo_common_mhm_mrm_variables::evalper, mo_common_variables::level1, mo_common_↩ constants::nodata_dp, and mo_read_nc::read_nc().

Referenced by mhm_driver().

Here is the call graph for this function:



Here is the caller graph for this function:



17.64 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.

17.64.1 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

17.64.2 Function/Subroutine Documentation

17.64.2.1 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

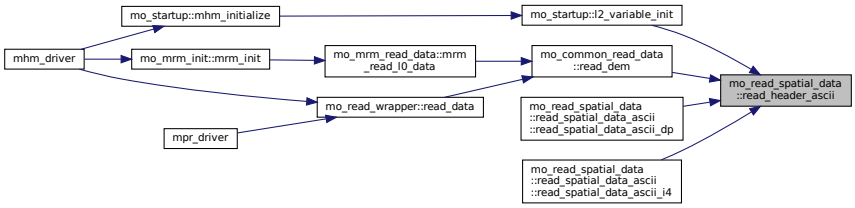
Date

Jan 2013

Definition at line 375 of file mo_read_spatial_data.f90.

Referenced by mo_startup::l2_variable_init(), mo_common_read_data::read_dem(), mo_read_spatial_data::read_spatial_data_ascii::read_spatial_data_ascii_dp(), and mo_read_spatial_data::read_spatial_data_ascii::read_spatial_data_ascii_i4().

Here is the caller graph for this function:



17.64.2.2 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

Definition at line 94 of file mo_read_spatial_data.f90.

17.64.2.3 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

Definition at line 234 of file mo_read_spatial_data.f90.

17.65 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.

17.65.1 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

17.65.2 Function/Subroutine Documentation

17.65.2.1 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

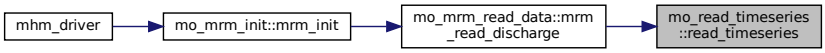
Date

Jan 2013

Definition at line 73 of file mo_read_timeseries.f90.

Referenced by mo_mrm_read_data::mrm_read_discharge().

Here is the caller graph for this function:



17.66 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.

17.66.1 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

17.66.2 Function/Subroutine Documentation

17.66.2.1 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

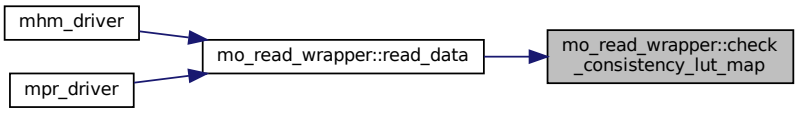
Date

Nov 2016

Definition at line 368 of file mo_read_wrapper.f90.

Referenced by read_data().

Here is the caller graph for this function:



17.66.2.2 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 domains to be modeled.

Parameters

in	*type(period), dimension(:), optional :: LAIPer*	

Authors

Juliane Mai & Matthias Zink

Date

Feb 2013

by default; when iFlag_soilDB = 0

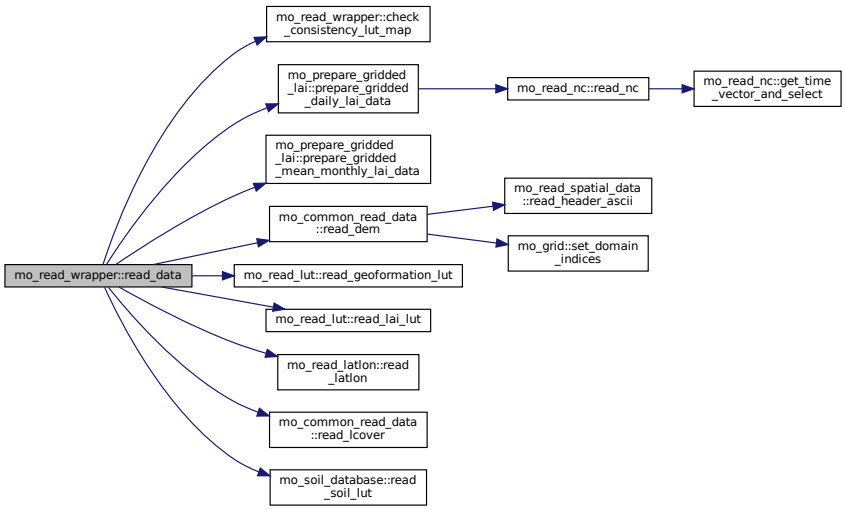
Definition at line 63 of file mo_read_wrapper.f90.

References check_consistency_lut_map(), mo_common_variables::dircommonfiles, mo_common_variables::dirmorpho, mo_common_variables::domainmeta, 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_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::laiboundaries, mo_mpr_global_variables::lailut, mo_mpr_global_variables::laiunitlist, mo_common_variables::level0, 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_

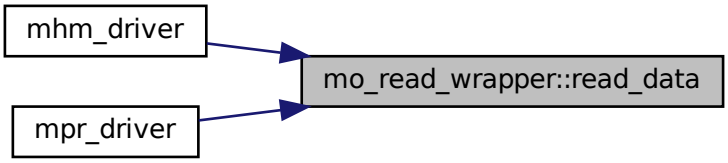
common_read_data::read_dem(), mo_read_lut::read_geoformation_lut(), mo_read_lut::read_lai_lut(), mo_read_↩ latlon::read_latlon(), mo_common_read_data::read_lcover(), mo_soil_database::read_soil_lut(), mo_mpr_global↩ _variables::soildb, mo_mpr_global_variables::timestep_lai_input, mo_mpr_file::uaspect, mo_mpr_file::ugeolut, mo_mpr_file::uhydrogeoclass, mo_mpr_file::ulaiclass, mo_mpr_file::ulailut, mo_mpr_file::uslope, and mo_mpr_↩ file::usoilclass.

Referenced by mhm_driver(), and mpr_driver().

Here is the call graph for this function:



Here is the caller graph for this function:



17.67 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 (OutFile)

  write restart files for each domain
- subroutine, public read_restart_states (iDomain, domainID, InFile)

  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)

17.67.1 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

17.67.2 Function/Subroutine Documentation

17.67.2.1 read_restart_states()

```
subroutine, public mo_restart::read_restart_states (
             integer(i4), intent(in) iDomain,
             integer(i4), intent(in) domainID,
             character(256), intent(in) InFile )
```

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) :: iDomain*	number of domains
in	*character(256) :: InFile*	Input Path including trailing slash

Authors

Stephan Thober

Date

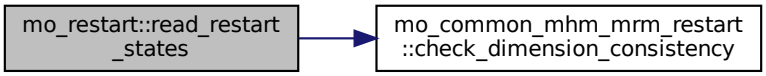
Apr 2013

Definition at line 344 of file mo_restart.f90.

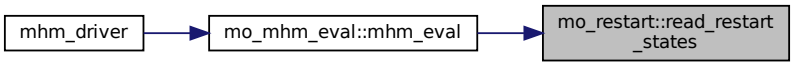
References mo_common_mhm_mrm_restart::check_dimension_consistency(), mo_mpr_global_variables::l1_aeroresist, mo_global_variables::l1_aetcanopy, mo_global_variables::l1_aetsealed, mo_global_variables::l1_aetsoil, mo_mpr_global_variables::l1_alpha, mo_global_variables::l1_baseflow, mo_mpr_global_variables::l1_degday, 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_constants::laivarname, mo_common_constants::landcoverperiodsvarname, 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, mo_common_variables::processmatrix, and mo_common_constants::soilhorizonsvarname.

Referenced by mo_mhm_eval::mhm_eval().

Here is the call graph for this function:



Here is the caller graph for this function:



17.67.2.2 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 )
```

Definition at line 830 of file mo_restart.f90.

17.67.2.3 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 )
```

Definition at line 785 of file mo_restart.f90.

17.67.2.4 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 )
```

Definition at line 875 of file mo_restart.f90.

17.67.2.5 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 )
```

Definition at line 932 of file mo_restart.f90.

17.67.2.6 write_restart_files()

```
subroutine, public mo_restart::write_restart_files (
            character(256), dimension(:), intent(in) OutFile )
```

write restart files for each domain

write restart files for each domain. For each domain three restart files are written. These are xxx_states.nc, xxx_↩ L11_config.nc, and xxx_config.nc (xxx being the three digit domain index). If a variable is added here, it should also be added in the read restart routines below.

Parameters

in	*character(256), dimension(:) :: OutFile*	Output Path for each domain

Authors

Stephan Thober

Date

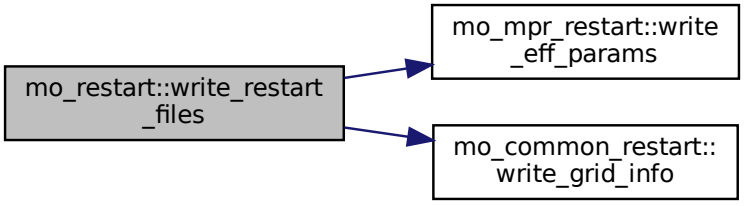
Jun 2014

Definition at line 99 of file mo_restart.f90.

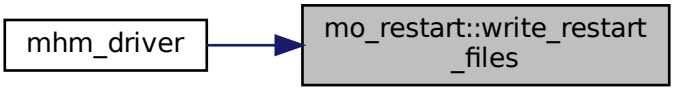
References mo_common_variables::domainmeta, mo_mpr_global_variables::horizondepth_mhm, mo_global↩ _variables::l1_aetcanopy, mo_global_variables::l1_aetsealed, mo_global_variables::l1_aetsoil, mo_global_↩ variables::l1_baseflow, mo_global_variables::l1_fastrunoff, mo_global_variables::l1_infilsoil, mo_global_variables↩ ::l1_inter, mo_global_variables::l1_melt, mo_global_variables::l1_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_mpr_global_variables::laiboundaries, 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_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:



17.68 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

17.68.1 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

17.68.2 Function/Subroutine Documentation

17.68.2.1 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 TS-1]
in	*REAL(dp) :: slow_interflow*	q_1 Slow runoff component [mm TS-1]
in	*REAL(dp) :: baseflow*	q_2 Baseflow [mm TS-1]
in	*REAL(dp) :: direct_runoff*	q_D Direct runoff from impervious areas [mm TS-1]
out	*REAL(dp) :: total_runoff*	q_T Generated runoff [mm TS-1]

Authors

Vladyslav Prykhodko

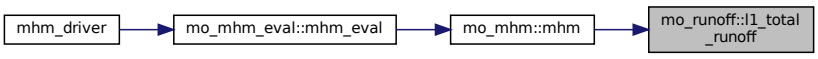
Date

Dec 2012

Definition at line 251 of file mo_runoff.f90.

Referenced by mo_mhm::mhm().

Here is the caller graph for this function:



17.68.2.2 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 [TS-1]

Parameters

in,out	*REAL(dp) :: sat_storage*	Groundwater storage [mm]
out	*REAL(dp) :: baseflow*	Baseflow [mm TS-1]

Authors

Vladyslav Prykhodko

Date

Dec 2012

Definition at line 194 of file mo_runoff.f90.

Referenced by mo_mhm::mhm().

Here is the caller graph for this function:

17.68.2.3 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 [TS-1]
in	*REAL(dp) :: kp*	Percolation coefficient [TS-1]
in	*REAL(dp) :: k0*	Recession coefficient of the upperreservoir, upper outlet [TS-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]

Parameters

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 TS-1]
out	*REAL(dp) :: fast_interflow*	Fast runoff component [mm TS-1]
out	*REAL(dp) :: perc*	Percolation [mm TS-1]

Authors

Vladyslav Prykhodko

Date

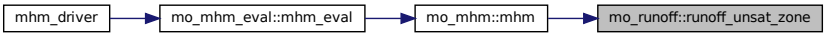
Dec 2012

Definition at line 77 of file mo_runoff.f90.

References mo_common_constants::eps_dp.

Referenced by mo_mhm::mhm().

Here is the caller graph for this function:



17.69 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.

17.69.1 Detailed Description

Snow melting and accumulation.

This module calculates snow melting and accumulation.

Authors

Vladyslav Prykhodko

Date

Dec 2012

17.69.2 Function/Subroutine Documentation

17.69.2.1 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 TS-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 TS-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

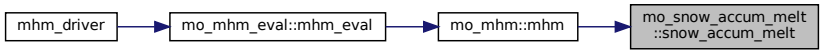
Date

Dec 2012

Definition at line 71 of file mo_snow_accum_melt.f90.

Referenced by mo_mhm::mhm().

Here is the caller graph for this function:



17.70 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.

17.70.1 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

17.70.2 Function/Subroutine Documentation

17.70.2.1 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

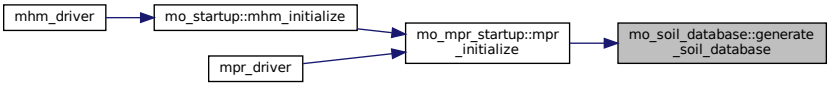
Dec 2012

Definition at line 380 of file mo_soil_database.f90.

References mo_mpr_global_variables::horizondepth_mhm, mo_mpr_global_variables::iflag_soildb, 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 caller graph for this function:



17.70.2.2 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

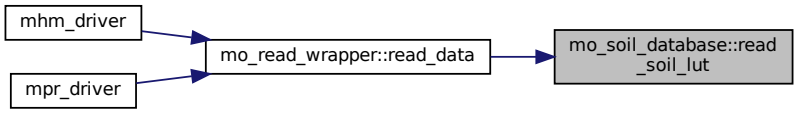
Dec 2012

Definition at line 61 of file mo_soil_database.f90.

References mo_common_constants::eps_dp, mo_mpr_global_variables::horizondepth_mhm, mo_mpr_global_↩ variables::iflag_soildb, 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 caller graph for this function:



17.71 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, public 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, public 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

17.71.1 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

17.71.2 Function/Subroutine Documentation

17.71.2.1 feddes_et_reduction()

```
elemental pure real(dp) function, public 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 )
```

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

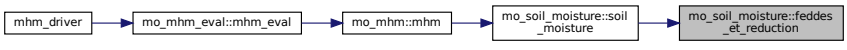
Date

March 2017

Definition at line 334 of file mo_soil_moisture.f90.

Referenced by soil_moisture().

Here is the caller graph for this function:



17.71.2.2 jarvis_et_reduction()

```
elemental pure real(dp) function, public 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 )
```

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

Definition at line 406 of file mo_soil_moisture.f90.

Referenced by soil_moisture().

Here is the caller graph for this function:

17.71.2.3 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(in) 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 TS-1]
in	*real(dp) :: pet*	Reference evapotranspiration [mm TS-1]
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 TS-1]

Parameters

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 TS-1]
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 TS-1]
out	*real(dp) :: aet_sealed*	actual ET from free-water surfaces,i.e impervious cover [mm TS-1]

Authors

Matthias Cuntz

Date

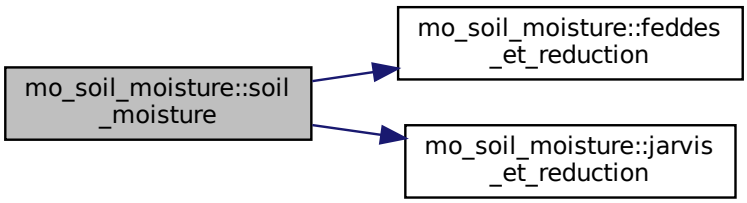
Dec 2012

Definition at line 95 of file mo_soil_moisture.f90.

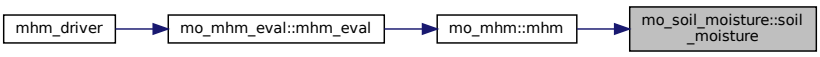
References mo_common_constants::eps_dp, feddes_et_reduction(), and jarvis_et_reduction().

Referenced by mo_mhm::mhm().

Here is the call graph for this function:



Here is the caller graph for this function:



17.72 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)

17.72.1 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

17.72.2 Function/Subroutine Documentation

17.72.2.1 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]
```

Definition at line 96 of file mo_spatial_agg_disagg_forcing.f90.

17.72.2.2 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 )
```

Definition at line 204 of file mo_spatial_agg_disagg_forcing.f90.

17.72.2.3 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 )
```

Definition at line 315 of file mo_spatial_agg_disagg_forcing.f90.

17.72.2.4 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 )
```

Definition at line 381 of file mo_spatial_agg_disagg_forcing.f90.

17.73 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 (iDomain, level0_iDomain, level2_iDomain)

  Initalize Level-2 meteorological forcings data.

17.73.1 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

17.73.2 Function/Subroutine Documentation

17.73.2.1 constants_init()

```
subroutine mo_startup::constants_init
```

Initialize mHM constants.

transformation of time units & initialize constants

Authors

Luis Samaniego

Date

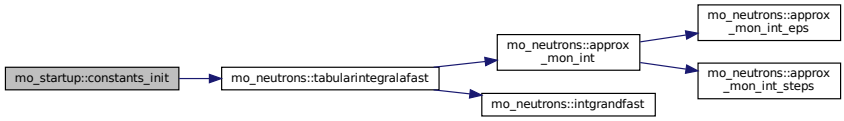
Dec 2012

Definition at line 144 of file mo_startup.f90.

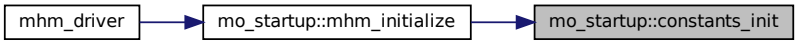
References mo_common_mhm_mrm_variables::c2tstu, mo_mpr_file::file_hydrogeoclass, mo_file::file_namelist_↩ mhm_param, mo_mpr_global_variables::geounitlist, mo_global_variables::neutron_integral_afast, mo_common_↩ variables::processmatrix, mo_neutrons::tabularintegralafast(), and mo_common_mhm_mrm_variables::timestep.

Referenced by mhm_initialize().

Here is the call graph for this function:



Here is the caller graph for this function:



17.73.2.2 l2_variable_init()

```
subroutine mo_startup::l2_variable_init (
            integer(i4), intent(in) iDomain,
```

```
type(grid), intent(in) level0_iDomain,
type(grid), intent(inout) level2_iDomain )
```

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) :: iDomain*	domain Id
in	*type(Grid) :: level0_iDomain*	
in,out	*type(Grid) :: level2_iDomain*	

Authors

Rohini Kumar

Date

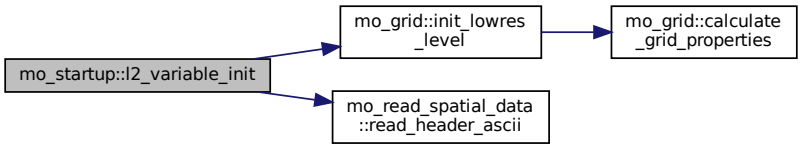
Feb 2013

Definition at line 211 of file mo_startup.f90.

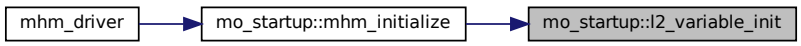
References mo_global_variables::dirprecipitation, mo_mpr_file::file_meteo_header, mo_grid::init_lowres_level(), mo_read_spatial_data::read_header_ascii(), and mo_mpr_file::umeteo_header.

Referenced by mhm_initialize().

Here is the call graph for this function:



Here is the caller graph for this function:



17.73.2.3 mhm_initialize()

```
subroutine, public mo_startup::mhm_initialize
```

Initialize main mHM variables.

Initialize main mHM variables for a given domain. 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

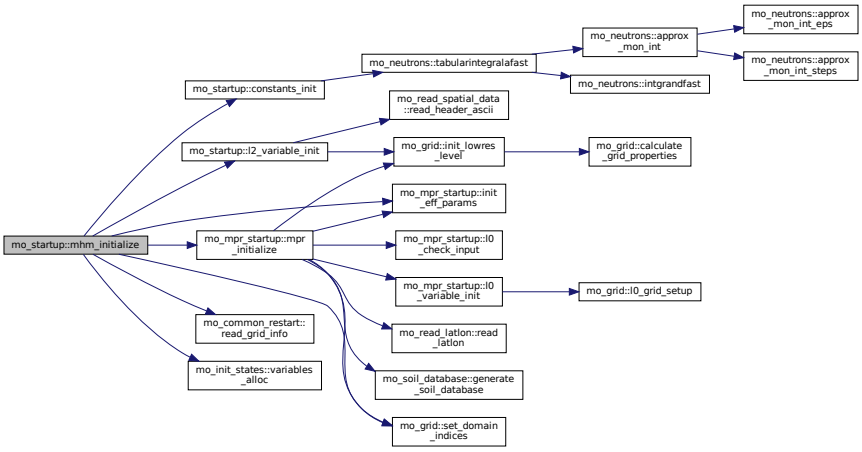
Dec 2012

Definition at line 70 of file mo_startup.f90.

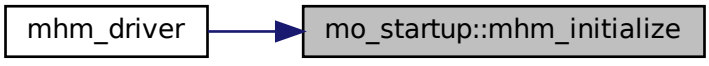
References constants_init(), mo_common_variables::domainmeta, mo_mpr_startup::init_eff_params(), l2_↩ variable_init(), mo_common_variables::level0, mo_common_variables::level1, mo_global_variables::level2, mo_↩ common_mhm_mrm_variables::mhmfilerestartin, mo_mpr_startup::mpr_initialize(), mo_common_restart::read_↩ grid_info(), mo_common_mhm_mrm_variables::read_restart, mo_grid::set_domain_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:



17.74 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

17.74.1 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

17.74.2 Function/Subroutine Documentation

17.74.2.1 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

Definition at line 122 of file mo_template.f90.

References pi_dp.

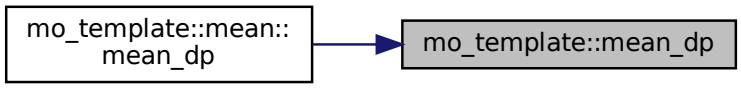
17.74.2.2 mean_dp()

```
real(dp) function mo_template::mean_dp (
             real(dp), dimension(:), intent(in) dat,
             logical, dimension(:), intent(in), optional mask )  [private]
```

Definition at line 137 of file mo_template.f90.

Referenced by mo_template::mean::mean_dp().

Here is the caller graph for this function:



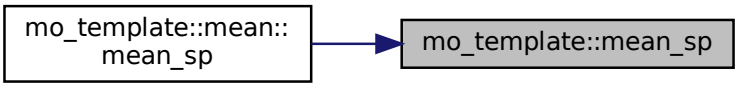
17.74.2.3 mean_sp()

```
real(sp) function mo_template::mean_sp (
             real(sp), dimension(:), intent(in) dat,
             logical, dimension(:), intent(in), optional mask )  [private]
```

Definition at line 167 of file mo_template.f90.

Referenced by mo_template::mean::mean_sp().

Here is the caller graph for this function:



17.74.3 Variable Documentation

17.74.3.1 itest

```
integer(i4), parameter mo_template::itest = 1  [private]
```

Definition at line 92 of file mo_template.f90.

17.74.3.2 pi_dp

```
real(dp), parameter, public mo_template::pi_dp = 3.14159265358979323846264338327950288419 7_dp
```

Constant Pi in double precision.

Definition at line 87 of file mo_template.f90.

Referenced by circum().

17.74.3.3 pi_sp

```
real(sp), parameter, public mo_template::pi_sp = 3.14159265358979323846264338327950288419 7_sp
```

Constant Pi in single precision.

Definition at line 89 of file mo_template.f90.

17.75 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.
- elemental subroutine, public temporal_disagg_meteo_weights (meteo_val_day, meteo_val_weights, meteo↩ _val, weights_correction)

Temporally distribute daily mean forcings onto time step.

- elemental subroutine, public temporal_disagg_flux_daynight (isday, ntimesteps_day, meteo_val_day, fday_↩ meteo_val, fnight_meteo_val, meteo_val)

  Temporally distribute daily mean forcings onto time step.

- elemental subroutine, public temporal_disagg_state_daynight (isday, ntimesteps_day, meteo_val_day, fday↩ _meteo_val, fnight_meteo_val, meteo_val, add_correction)

  Temporally distribute daily mean state forcings onto time step.

17.75.1 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

17.75.2 Function/Subroutine Documentation

17.75.2.1 temporal_disagg_flux_daynight()

```
elemental subroutine, public mo_temporal_disagg_forcing::temporal_disagg_flux_daynight (
             logical, intent(in) isday,
             real(dp), intent(in) ntimesteps_day,
             real(dp), intent(in) meteo_val_day,
             real(dp), intent(in) fday_meteo_val,
             real(dp), intent(in) fnight_meteo_val,
             real(dp), intent(out) meteo_val )
```

Temporally distribute daily mean forcings onto time step.

Calculates actual meteo forcings from daily mean inputs. They are distributed with predefined factors/summands for day and night.

Parameters

in	*logical :: isday*	is day or night
in	*real(dp) :: ntimesteps_day*	# of time steps per day
in	*real(dp) :: meteo_val_day*	Daily mean meteo value
in	*real(dp) :: fday_meteo_val*	Daytime fraction of meteo value
in	*real(dp) :: fnight_meteo_val*	Nighttime fraction of meteo value
out	*real(dp) :: meteo_val*	Actual meteo value

Authors

Sebastian Mueller

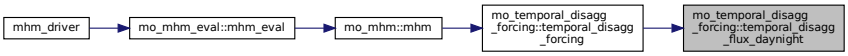
Date

Jun 2020

Definition at line 222 of file mo_temporal_disagg_forcing.f90.

Referenced by temporal_disagg_forcing().

Here is the caller graph for this function:



17.75.2.2 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/d]
in	*real(dp) :: pet_day*	Daily mean ET [mm/d]
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
in	*real(dp) :: fday_temp*	Daytime air temparture increase
in	*real(dp) :: fnight_prec*	Daytime fraction of precipitation

Parameters

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/d]
out	*real(dp) :: pet*	Reference ET [mm/d]
out	*real(dp) :: temp*	Air temperature [K]

Authors

Matthias Cuntz

Date

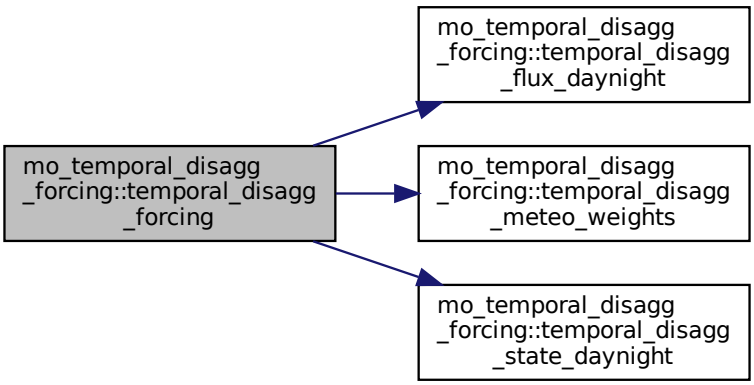
Dec 2012

Definition at line 77 of file mo_temporal_disagg_forcing.f90.

References temporal_disagg_flux_daynight(), temporal_disagg_meteo_weights(), and temporal_disagg_state_↩ daynight().

Referenced by mo_mhm::mhm().

Here is the call graph for this function:



Here is the caller graph for this function:

17.75.2.3 temporal_disagg_meteo_weights()

```
elemental subroutine, public mo_temporal_disagg_forcing::temporal_disagg_meteo_weights (
            real(dp), intent(in) meteo_val_day,
            real(dp), intent(in) meteo_val_weights,
            real(dp), intent(out) meteo_val,
            real(dp), intent(in), optional weights_correction )
```

Temporally distribute daily mean forcings onto time step.

Calculates actual meteo forcings from daily mean inputs. They are distributed with predefined weights onto the day.

Parameters

in	*logical :: isday*	is day or night
in	*real(dp) :: ntimesteps_day*	# of time steps per day
in	*real(dp) :: meteo_val_day*	Daily mean meteo value
out	*real(dp) :: meteo_val*	Actual meteo value
in	*real(dp), optional :: weights_correction*	Additive correction value before weights are applied

Authors

Sebastian Mueller

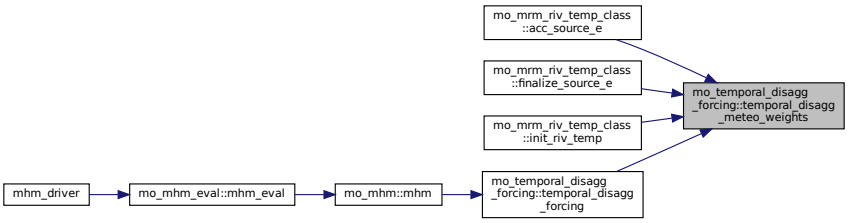
Date

Jun 2020

Definition at line 167 of file mo_temporal_disagg_forcing.f90.

Referenced by mo_mrm_riv_temp_class::acc_source_e(), mo_mrm_riv_temp_class::finalize_source_e(), mo_↩ mrm_riv_temp_class::init_riv_temp(), and temporal_disagg_forcing().

Here is the caller graph for this function:



17.75.2.4 temporal_disagg_state_daynight()

```
elemental subroutine, public mo_temporal_disagg_forcing::temporal_disagg_state_daynight (
            logical, intent(in) isday,
            real(dp), intent(in) ntimesteps_day,
            real(dp), intent(in) meteo_val_day,
```

```
real(dp), intent(in) fday_meteo_val,
real(dp), intent(in) fnight_meteo_val,
real(dp), intent(out) meteo_val,
logical, intent(in), optional add_correction )
```

Temporally distribute daily mean state forcings onto time step.

Calculates meteo forcings from daily mean inputs of a state variable. They are distributed with predefined factors/summands for day and night.

Parameters

in	*logical :: isday*	is day or night
in	*real(dp) :: ntimesteps_day*	# of time steps per day
in	*real(dp) :: meteo_val_day*	Daily mean meteo value
in	*real(dp) :: fday_meteo_val*	Daytime fraction/addition of meteo value
in	*real(dp) :: fnight_meteo_val*	Nighttime fraction/addition of meteo value
out	*real(dp) :: meteo_val*	Actual meteo value
in	*logical, optional :: add_correction*	State if correction should be added instead of multiplied

Authors

Sebastian Mueller

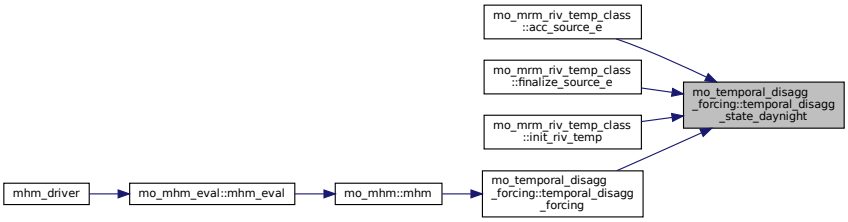
Date

Jun 2020

Definition at line 288 of file mo_temporal_disagg_forcing.f90.

Referenced by mo_mrm_riv_temp_class::acc_source_e(), mo_mrm_riv_temp_class::finalize_source_e(), mo_mrm_riv_temp_class::init_riv_temp(), and temporal_disagg_forcing().

Here is the caller graph for this function:



17.76 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

17.76.1 Detailed Description

Module containing upscaling operators.

This module provides the routines for upscaling_operators.

Authors

Giovanni Dalmasso, Rohini Kumar

Date

Dec 2012

17.76.2 Function/Subroutine Documentation

17.76.2.1 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

Definition at line 153 of file mo_upscaling_operators.f90.

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:

17.76.2.2 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

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	*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

Definition at line 66 of file mo_upscaling_operators.f90.

17.76.2.3 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

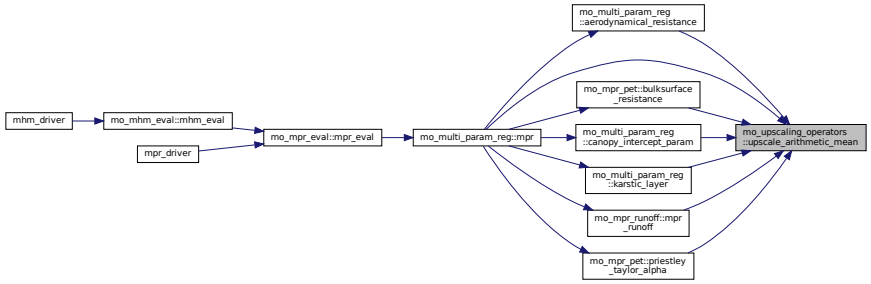
Date

Dec 2012

Definition at line 267 of file mo_upscaling_operators.f90.

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:



17.76.2.4 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

Definition at line 470 of file mo_upscaling_operators.f90.

17.76.2.5 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

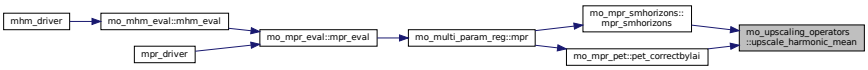
Date

Dec 2012

Definition at line 370 of file mo_upscaling_operators.f90.

Referenced by mo_mpr_smhorizons::mpr_smhorizons(), and mo_mpr_pet::pet_correctbylai().

Here is the caller graph for this function:



17.76.2.6 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

Definition at line 577 of file mo_upscaling_operators.f90.

17.77 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.

17.77.1 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

17.77.2 Function/Subroutine Documentation

17.77.2.1 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

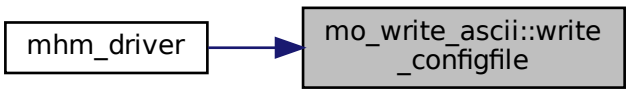
Jun 2018

Definition at line 85 of file mo_write_ascii.f90.

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_global_variables::dirtemperature, mo_common_variables::domainmeta, 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_common_variables::iflag_cordinate_sys, mo_mrm_global_variables::inflowgauge, 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_common_variables::mhmfilerestartout, mo_mrm_global_variables::ngaugeslocal, 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 caller graph for this function:



17.77.2.2 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

Definition at line 425 of file mo_write_ascii.f90.

References mo_common_variables::dirconfigout, mo_common_mhm_mrm_file::file_opti, and mo_common_↩ mhm_mrm_file::uopti.

17.77.2.3 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

Definition at line 512 of file mo_write_ascii.f90.

References mo_common_variables::dirconfigout, mo_common_mhm_mrm_file::file_opti_nml, mo_common_↩ variables::nprocesses, and mo_common_mhm_mrm_file::uopti_nml.

17.78 mo_write_fluxes_states Module Reference

Creates NetCDF output for different fluxes and state variables of mHM.

Data Types

- interface outputvariable
- interface outputdataset

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 (iDomain, mask1, nCells)

  Initialize OutputDataset.
- subroutine updatedataset (self, sidx, eidx, L1_fSealed, L1_fNotSealed, L1_inter, L1_snowPack, L1_soil↩ Moist, 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 (iDomain)

  Create and initialize output file for X & Y coordinate system.
- subroutine writevariableattributes (var, long_name, unit)

  Write output variable attributes.
- character(16) function fluxesunit (iDomain)

  Generate a unit string.

17.78.1 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

17.78.2 Function/Subroutine Documentation

17.78.2.1 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

Definition at line 851 of file mo_write_fluxes_states.f90.

References mo_common_variables::dirout.

17.78.2.2 createoutputfile()

```
type(ncdataset) function mo_write_fluxes_states::createoutputfile (
             integer(i4), intent(in) iDomain )
```

Create and initialize output file for X & Y coordinate system.

Create output file, write all non-dynamic variables and global attributes for the given domain for X & Y coordinate system

Returns

type(NcDataset)

Parameters

in	*integer(i4) :: iDomain*	-> domain id

Authors

David Schaefer

Date

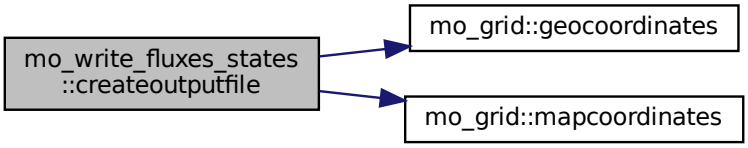
June 2015

Definition at line 893 of file mo_write_fluxes_states.f90.

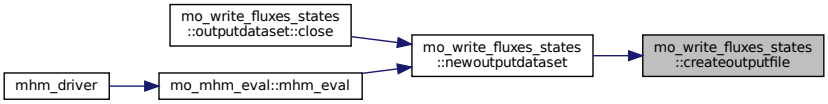
References mo_common_variables::dirout, mo_common_mhm_mrm_variables::evalper, mo_grid::geocoordinates(), mo_common_variables::iflag_cordinate_sys, mo_common_variables::level1, mo_grid::mapcoordinates(), mo_common_constants::nodata_dp, mo_global_variables::output_double_precision, and mo_file::version.

Referenced by newoutputdataset().

Here is the call graph for this function:



Here is the caller graph for this function:



17.78.2.3 fluxesunit()

```
character(16) function mo_write_fluxes_states::fluxesunit (
            integer(i4), intent(in) iDomain )  [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) :: iDomain*	

Authors

David Schaefer

Date

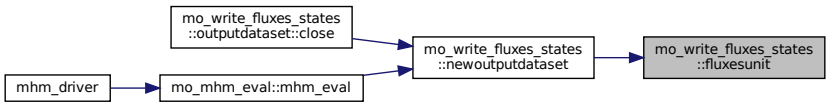
June 2015

Definition at line 1089 of file mo_write_fluxes_states.f90.

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:



17.78.2.4 newoutputdataset()

```
type(outputdataset) function, public mo_write_fluxes_states::newoutputdataset (
            integer(i4), intent(in) iDomain,
            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) :: iDomain*	-> domain id
in	*logical, dimension(:, :) :: mask1*	-> L1 mask to reconstruct the data
in	*integer(i4) :: nCells*	

Authors

Matthias Zink

Date

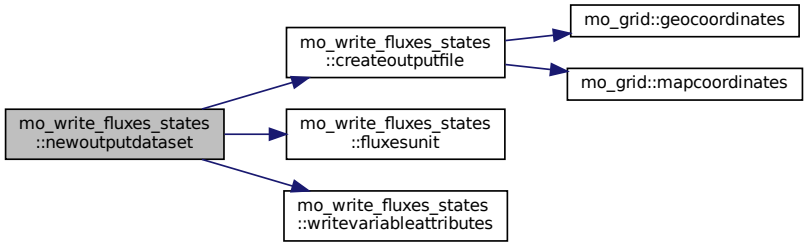
Apr 2013

Definition at line 242 of file mo_write_fluxes_states.f90.

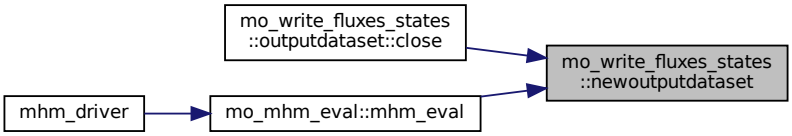
References createoutputfile(), fluxesunit(), mo_common_variables::iflag_cordinate_sys, mo_mpr_global_variables::nsoilhorizons_mhm, mo_global_variables::outputflxstate, and writevariableattributes().

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:



17.78.2.5 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 domain

Parameters

in	*logical, dimension(:, :) :: mask*	
in	*logical, optional :: avg*	-> average the data before writing

Authors

David Schaefer

Date

June 2015

Definition at line 103 of file mo_write_fluxes_states.f90.

References mo_global_variables::output_deflate_level.

17.78.2.6 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*	

Parameters

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

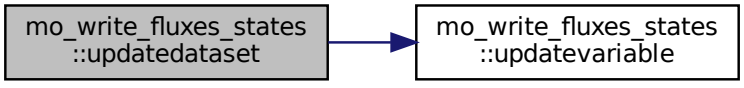
Date

Apr 2013

Definition at line 520 of file mo_write_fluxes_states.f90.

References mo_mpr_global_variables::nsoilhorizons_mhm, mo_global_variables::outputflxstate, and updatevariable().

Here is the call graph for this function:



17.78.2.7 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

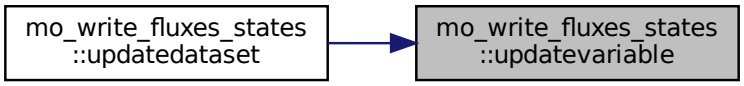
Date

June 2015

Definition at line 159 of file mo_write_fluxes_states.f90.

Referenced by updatedataset().

Here is the caller graph for this function:



17.78.2.8 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

Definition at line 808 of file mo_write_fluxes_states.f90.

17.78.2.9 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

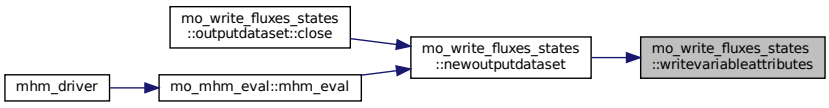
June 2015

Definition at line 1048 of file mo_write_fluxes_states.f90.

References mo_common_constants::nodata_dp.

Referenced by newoutputdataset().

Here is the caller graph for this function:



17.78.2.10 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

Definition at line 196 of file mo_write_fluxes_states.f90.

References mo_common_constants::nodata_dp.

Chapter 18

Data Type Documentation

18.1 mo_common_mhm_mrm_restart::check_consistency_element Interface Reference

Private Member Functions

- subroutine check_consistency_element_i4 (item1, item2, name, iBasin)
- subroutine check_consistency_element_dp (item1, item2, name, iBasin)

18.1.1 Detailed Description

Definition at line 21 of file mo_common_mHM_mRM_restart.f90.

18.1.2 Member Function/Subroutine Documentation

18.1.2.1 check_consistency_element_dp()

```
subroutine mo_common_mhm_mrm_restart::check_consistency_element::check_consistency_element_dp
(
            real(dp), intent(in) item1,
            real(dp), intent(in) item2,
            character(*), intent(in) name,
            integer(i4), intent(in) iBasin )  [private]
```

Definition at line 90 of file mo_common_mHM_mRM_restart.f90.

18.1.2.2 check_consistency_element_i4()

```
subroutine mo_common_mhm_mrm_restart::check_consistency_element::check_consistency_element_i4
(
            integer(i4), intent(in) item1,
            integer(i4), intent(in) item2,
            character(*), intent(in) name,
            integer(i4), intent(in) iBasin )  [private]
```

Definition at line 109 of file mo_common_mHM_mRM_restart.f90.

The documentation for this interface was generated from the following file:

- src/common_mHM_mRM/mo_common_mHM_mRM_restart.f90

18.2 mo_common_datetime_type::datetimeinfo Type Reference

Private Member Functions

- procedure init => datetimeinfo_init
- procedure increment => datetimeinfo_increment
- procedure update_lai_timestep => datetimeinfo_update_lai_timestep
- procedure writeout => datetimeinfo_writeout

Private Attributes

- integer(i4) ntimesteps

  number of timesteps in simulation period
- real(dp) newtime

  starts with simPer(iDomain)julStart, then increments with julday(...)
- integer(i4) day

  current day
- integer(i4) month

  current month
- integer(i4) year

  current year
- integer(i4), private hour
- integer(i4), private prev_day
- integer(i4), private prev_month
- integer(i4), private prev_year
- logical is_new_day
- logical is_new_month
- logical is_new_year
- integer(i4) ilai
- integer(i4) yid
- integer(i4) tindex_out

18.2.1 Detailed Description

Definition at line 27 of file mo_common_datetime_type.f90.

18.2.2 Member Function/Subroutine Documentation

18.2.2.1 increment()

```
procedure mo_common_datetime_type::datetimeinfo::increment  [private]
```

Definition at line 62 of file mo_common_datetime_type.f90.

18.2.2.2 init()

```
procedure mo_common_datetime_type::datetimeinfo::init  [private]
```

Definition at line 61 of file mo_common_datetime_type.f90.

18.2.2.3 update_lai_timestep()

```
procedure mo_common_datetime_type::datetimeinfo::update_lai_timestep  [private]
```

Definition at line 63 of file mo_common_datetime_type.f90.

18.2.2.4 writeout()

```
procedure mo_common_datetime_type::datetimeinfo::writeout  [private]
```

Definition at line 65 of file mo_common_datetime_type.f90.

18.2.3 Member Data Documentation

18.2.3.1 day

```
integer(i4) mo_common_datetime_type::datetimeinfo::day  [private]
```

current day

Definition at line 35 of file mo_common_datetime_type.f90.

18.2.3.2 hour

```
integer(i4), private mo_common_datetime_type::datetimeinfo::hour  [private]
```

Definition at line 41 of file mo_common_datetime_type.f90.

18.2.3.3 ilai

```
integer(i4) mo_common_datetime_type::datetimeinfo::ilai  [private]
```

Definition at line 54 of file mo_common_datetime_type.f90.

18.2.3.4 is_new_day

```
logical mo_common_datetime_type::datetimeinfo::is_new_day  [private]
```

Definition at line 50 of file mo_common_datetime_type.f90.

18.2.3.5 is_new_month

```
logical mo_common_datetime_type::datetimeinfo::is_new_month  [private]
```

Definition at line 51 of file mo_common_datetime_type.f90.

18.2.3.6 is_new_year

```
logical mo_common_datetime_type::datetimeinfo::is_new_year  [private]
```

Definition at line 52 of file mo_common_datetime_type.f90.

18.2.3.7 month

```
integer(i4) mo_common_datetime_type::datetimeinfo::month  [private]
```

current month

Definition at line 37 of file mo_common_datetime_type.f90.

18.2.3.8 newtime

```
real(dp) mo_common_datetime_type::datetimeinfo::newtime  [private]
```

starts with simPer(iDomain)julStart, then increments with julday(...)

Definition at line 32 of file mo_common_datetime_type.f90.

18.2.3.9 ntimesteps

```
integer(i4) mo_common_datetime_type::datetimeinfo::ntimesteps  [private]
```

number of timesteps in simulation period

Definition at line 29 of file mo_common_datetime_type.f90.

18.2.3.10 prev_day

```
integer(i4), private mo_common_datetime_type::datetimeinfo::prev_day  [private]
```

Definition at line 45 of file mo_common_datetime_type.f90.

18.2.3.11 prev_month

```
integer(i4), private mo_common_datetime_type::datetimeinfo::prev_month  [private]
```

Definition at line 46 of file mo_common_datetime_type.f90.

18.2.3.12 prev_year

```
integer(i4), private mo_common_datetime_type::datetimeinfo::prev_year  [private]
```

Definition at line 47 of file mo_common_datetime_type.f90.

18.2.3.13 tindex_out

```
integer(i4) mo_common_datetime_type::datetimeinfo::tindex_out  [private]
```

Definition at line 58 of file mo_common_datetime_type.f90.

18.2.3.14 year

```
integer(i4) mo_common_datetime_type::datetimeinfo::year  [private]
```

current year

Definition at line 39 of file mo_common_datetime_type.f90.

18.2.3.15 yid

```
integer(i4) mo_common_datetime_type::datetimeinfo::yid  [private]
```

Definition at line 55 of file mo_common_datetime_type.f90.

The documentation for this type was generated from the following file:

- src/common/mo_common_datetime_type.f90

18.3 mo_common_variables::domain_meta Type Reference

Public Attributes

- integer(i4) ndomains
- integer(i4) overallnumberofdomains
- integer(i4), dimension(:), allocatable indices
- integer(i4), dimension(:), allocatable l0datafrom
- integer(i4), dimension(:), allocatable optidata
- logical, dimension(:), allocatable dorouting
- logical ismasterincomlocal
- type(mpi_comm) commaster
- type(mpi_comm) comlocal

18.3.1 Detailed Description

Definition at line 138 of file mo_common_variables.f90.

18.3.2 Member Data Documentation

18.3.2.1 comlocal

```
type(mpi_comm) mo_common_variables::domain_meta::comlocal
```

Definition at line 157 of file mo_common_variables.f90.

18.3.2.2 commaster

```
type(mpi_comm) mo_common_variables::domain_meta::commaster
```

Definition at line 155 of file mo_common_variables.f90.

18.3.2.3 dorouting

```
logical, dimension(:), allocatable mo_common_variables::domain_meta::dorouting
```

Definition at line 152 of file mo_common_variables.f90.

18.3.2.4 indices

```
integer(i4), dimension(:), allocatable mo_common_variables::domain_meta::indices
```

Definition at line 141 of file mo_common_variables.f90.

18.3.2.5 ismasterincomlocal

```
logical mo_common_variables::domain_meta::ismasterincomlocal
```

Definition at line 154 of file mo_common_variables.f90.

18.3.2.6 l0datafrom

```
integer(i4), dimension(:), allocatable mo_common_variables::domain_meta::l0datafrom
```

Definition at line 142 of file mo_common_variables.f90.

18.3.2.7 ndomains

```
integer(i4) mo_common_variables::domain_meta::ndomains
```

Definition at line 139 of file mo_common_variables.f90.

18.3.2.8 optidata

```
integer(i4), dimension(:), allocatable mo_common_variables::domain_meta::optidata
```

Definition at line 151 of file mo_common_variables.f90.

18.3.2.9 overallnumberofdomains

```
integer(i4) mo_common_variables::domain_meta::overallnumberofdomains
```

Definition at line 140 of file mo_common_variables.f90.

The documentation for this type was generated from the following file:

- src/common/mo_common_variables.f90

18.4 mo_mrm_global_variables::domaininfo_mrm Type Reference

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

18.4.1 Detailed Description

Definition at line 93 of file mo_mrm_global_variables.f90.

18.4.2 Member Data Documentation

18.4.2.1 gaugeidlist

```
integer(i4), dimension(:), allocatable mo_mrm_global_variables::domaininfo_mrm::gaugeidlist
```

Definition at line 97 of file mo_mrm_global_variables.f90.

18.4.2.2 gaugeindexlist

```
integer(i4), dimension(:), allocatable mo_mrm_global_variables::domaininfo_mrm::gaugeindexlist
```

Definition at line 98 of file mo_mrm_global_variables.f90.

18.4.2.3 gaugenodelist

```
integer(i4), dimension(:), allocatable mo_mrm_global_variables::domaininfo_mrm::gaugenodelist
```

Definition at line 99 of file mo_mrm_global_variables.f90.

18.4.2.4 inflowgaugeheadwater

```
logical, dimension(:), allocatable mo_mrm_global_variables::domaininfo_mrm::inflowgaugeheadwater
```

Definition at line 106 of file mo_mrm_global_variables.f90.

18.4.2.5 inflowgaugeidlist

```
integer(i4), dimension(:), allocatable mo_mrm_global_variables::domaininfo_mrm::inflowgaugeidlist
```

Definition at line 103 of file mo_mrm_global_variables.f90.

18.4.2.6 inflowgaugeindexlist

```
integer(i4), dimension(:), allocatable mo_mrm_global_variables::domaininfo_mrm::inflowgaugeindexlist
```

Definition at line 104 of file mo_mrm_global_variables.f90.

18.4.2.7 inflowgaugenodelist

```
integer(i4), dimension(:), allocatable mo_mrm_global_variables::domaininfo_mrm::inflowgaugenodelist
```

Definition at line 105 of file mo_mrm_global_variables.f90.

18.4.2.8 l0_coloutlet

```
integer(i4), dimension(:), allocatable mo_mrm_global_variables::domaininfo_mrm::l0_coloutlet
```

Definition at line 112 of file mo_mrm_global_variables.f90.

18.4.2.9 l0_noutlet

```
integer(i4) mo_mrm_global_variables::domaininfo_mrm::l0_noutlet
```

Definition at line 110 of file mo_mrm_global_variables.f90.

18.4.2.10 l0_rowoutlet

```
integer(i4), dimension(:), allocatable mo_mrm_global_variables::domaininfo_mrm::l0_rowoutlet
```

Definition at line 111 of file mo_mrm_global_variables.f90.

18.4.2.11 ngauges

```
integer(i4) mo_mrm_global_variables::domaininfo_mrm::ngauges
```

Definition at line 96 of file mo_mrm_global_variables.f90.

18.4.2.12 ninflowgauges

```
integer(i4) mo_mrm_global_variables::domaininfo_mrm::ninflowgauges
```

Definition at line 102 of file mo_mrm_global_variables.f90.

The documentation for this type was generated from the following file:

- src/mRM/mo_mrm_global_variables.f90

18.5 mo_mrm_global_variables::gaugingstation Type Reference

Public Attributes

- integer(i4), dimension(:), allocatable domainid
- integer(i4), dimension(:), allocatable gaugeid
- character(256), dimension(:), allocatable fname
- real(dp), dimension(:, :), allocatable q
- real(dp), dimension(:, :), allocatable t

18.5.1 Detailed Description

Definition at line 79 of file mo_mrm_global_variables.f90.

18.5.2 Member Data Documentation

18.5.2.1 domainid

```
integer(i4), dimension(:), allocatable mo_mrm_global_variables::gaugingstation::domainid
```

Definition at line 80 of file mo_mrm_global_variables.f90.

18.5.2.2 fname

```
character(256), dimension(:), allocatable mo_mrm_global_variables::gaugingstation::fname
```

Definition at line 82 of file mo_mrm_global_variables.f90.

18.5.2.3 gaugeid

```
integer(i4), dimension(:), allocatable mo_mrm_global_variables::gaugingstation::gaugeid
```

Definition at line 81 of file mo_mrm_global_variables.f90.

18.5.2.4 q

```
real(dp), dimension(:, :), allocatable mo_mrm_global_variables::gaugingstation::q
```

Definition at line 83 of file mo_mrm_global_variables.f90.

18.5.2.5 t

```
real(dp), dimension(:, :), allocatable mo_mrm_global_variables::gaugingstation::t
```

Definition at line 85 of file mo_mrm_global_variables.f90.

The documentation for this type was generated from the following file:

- src/mRM/mo_mrm_global_variables.f90

18.6 mo_common_variables::grid Type Reference

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

18.6.1 Detailed Description

Definition at line 76 of file mo_common_variables.f90.

18.6.2 Member Data Documentation

18.6.2.1 cellarea

```
real(dp), dimension(:), allocatable mo_common_variables::grid::cellarea
```

Definition at line 93 of file mo_common_variables.f90.

18.6.2.2 cellcoor

```
integer(i4), dimension(:, :), allocatable mo_common_variables::grid::cellcoor
```

Definition at line 92 of file mo_common_variables.f90.

18.6.2.3 cellsize

```
real(dp) mo_common_variables::grid::cellsize
```

Definition at line 83 of file mo_common_variables.f90.

18.6.2.4 id

```
integer(i4), dimension(:), allocatable mo_common_variables::grid::id
```

Definition at line 94 of file mo_common_variables.f90.

18.6.2.5 iend

```
integer(i4) mo_common_variables::grid::iend
```

Definition at line 90 of file mo_common_variables.f90.

18.6.2.6 istart

```
integer(i4) mo_common_variables::grid::istart
```

Definition at line 89 of file mo_common_variables.f90.

18.6.2.7 mask

```
logical, dimension(:, :), allocatable mo_common_variables::grid::mask
```

Definition at line 87 of file mo_common_variables.f90.

18.6.2.8 ncells

```
integer(i4) mo_common_variables::grid::ncells
```

Definition at line 80 of file mo_common_variables.f90.

18.6.2.9 ncols

```
integer(i4) mo_common_variables::grid::ncols
```

Definition at line 78 of file mo_common_variables.f90.

18.6.2.10 nodata_value

```
real(dp) mo_common_variables::grid::nodata_value
```

Definition at line 84 of file mo_common_variables.f90.

18.6.2.11 nrows

```
integer(i4) mo_common_variables::grid::nrows
```

Definition at line 79 of file mo_common_variables.f90.

18.6.2.12 x

```
real(dp), dimension(:, :), allocatable mo_common_variables::grid::x
```

Definition at line 85 of file mo_common_variables.f90.

18.6.2.13 xllcorner

```
real(dp) mo_common_variables::grid::xllcorner
```

Definition at line 81 of file mo_common_variables.f90.

18.6.2.14 y

```
real(dp), dimension(:, :), allocatable mo_common_variables::grid::y
```

Definition at line 86 of file mo_common_variables.f90.

18.6.2.15 yllcorner

```
real(dp) mo_common_variables::grid::yllcorner
```

Definition at line 82 of file mo_common_variables.f90.

The documentation for this type was generated from the following file:

- src/common/mo_common_variables.f90

18.7 mo_common_variables::gridremapper Type Reference

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

18.7.1 Detailed Description

Definition at line 101 of file mo_common_variables.f90.

18.7.2 Member Data Documentation

18.7.2.1 high_res_grid

```
type(grid), pointer mo_common_variables::gridremapper::high_res_grid
```

Definition at line 102 of file mo_common_variables.f90.

18.7.2.2 left_bound

```
integer(i4), dimension(:), allocatable mo_common_variables::gridremapper::left_bound
```

Definition at line 108 of file mo_common_variables.f90.

18.7.2.3 low_res_grid

```
type(grid), pointer mo_common_variables::gridremapper::low_res_grid
```

Definition at line 103 of file mo_common_variables.f90.

18.7.2.4 lower_bound

```
integer(i4), dimension(:), allocatable mo_common_variables::gridremapper::lower_bound
```

Definition at line 106 of file mo_common_variables.f90.

18.7.2.5 lowres_id_on_highres

```
integer(i4), dimension(:, :), allocatable mo_common_variables::gridremapper::lowres_id_on_↩
highres
```

Definition at line 111 of file mo_common_variables.f90.

18.7.2.6 n_subcells

```
integer(i4), dimension(:), allocatable mo_common_variables::gridremapper::n_subcells
```

Definition at line 110 of file mo_common_variables.f90.

18.7.2.7 right_bound

```
integer(i4), dimension(:), allocatable mo_common_variables::gridremapper::right_bound
```

Definition at line 109 of file mo_common_variables.f90.

18.7.2.8 upper_bound

```
integer(i4), dimension(:), allocatable mo_common_variables::gridremapper::upper_bound
```

Definition at line 107 of file mo_common_variables.f90.

The documentation for this type was generated from the following file:

- src/common/mo_common_variables.f90

18.8 mo_template::mean Interface Reference

The average.

Public Member Functions

- real(sp) function mean_sp (dat, mask)
- real(dp) function mean_dp (dat, mask)

18.8.1 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

Definition at line 79 of file mo_template.f90.

18.8.2 Member Function/Subroutine Documentation

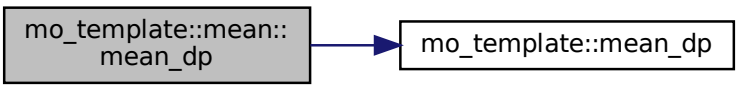
18.8.2.1 mean_dp()

```
real(dp) function mo_template::mean::mean_dp (
             real(dp), dimension(:), intent(in) dat,
             logical, dimension(:), intent(in), optional mask )
```

Definition at line 137 of file mo_template.f90.

References mo_template::mean_dp().

Here is the call graph for this function:



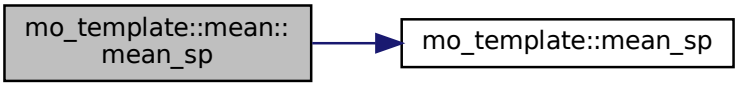
18.8.2.2 mean_sp()

```
real(sp) function mo_template::mean::mean_sp (
             real(sp), dimension(:), intent(in) dat,
             logical, dimension(:), intent(in), optional mask )
```

Definition at line 167 of file mo_template.f90.

References mo_template::mean_sp().

Here is the call graph for this function:



The documentation for this interface was generated from the following file:

- src/common/mo_template.f90

18.9 mo_mrm_write_fluxes_states::outputdataset Interface Reference

Public Member Functions

- procedure, public updatedataset
- procedure, public writetimestep
- procedure, public close

Public Attributes

- integer(i4) idomain
- integer(i4) domain
- 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

18.9.1 Detailed Description

Definition at line 45 of file mo_mrm_write_fluxes_states.f90.

18.9.2 Member Function/Subroutine Documentation

18.9.2.1 close()

```
procedure, public mo_mrm_write_fluxes_states::outputdataset::close
```

Definition at line 54 of file mo_mrm_write_fluxes_states.f90.

18.9.2.2 updatedataset()

```
procedure, public mo_mrm_write_fluxes_states::outputdataset::updatedataset
```

Definition at line 52 of file mo_mrm_write_fluxes_states.f90.

18.9.2.3 writetimestep()

```
procedure, public mo_mrm_write_fluxes_states::outputdataset::writetimestep
```

Definition at line 53 of file mo_mrm_write_fluxes_states.f90.

18.9.3 Member Data Documentation

18.9.3.1 all

```
type(outputvariable), allocatable mo_mrm_write_fluxes_states::outputdataset::all
```

Definition at line 48 of file mo_mrm_write_fluxes_states.f90.

18.9.3.2 count

```
integer(i4) mo_mrm_write_fluxes_states::outputdataset::count
```

Definition at line 49 of file mo_mrm_write_fluxes_states.f90.

18.9.3.3 counter

```
integer(i4) mo_mrm_write_fluxes_states::outputdataset::counter = 0
```

Definition at line 49 of file mo_mrm_write_fluxes_states.f90.

18.9.3.4 created

```
type(outputvariable), dimension (dynamic), allocatable mo_mrm_write_fluxes_states::outputdataset↩
::created
```

Definition at line 48 of file mo_mrm_write_fluxes_states.f90.

18.9.3.5 domain

```
integer(i4) mo_mrm_write_fluxes_states::outputdataset::domain
```

Definition at line 46 of file mo_mrm_write_fluxes_states.f90.

18.9.3.6 id

```
integer(i4) mo_mrm_write_fluxes_states::outputdataset::id
```

Definition at line 46 of file mo_mrm_write_fluxes_states.f90.

18.9.3.7 idomain

```
integer(i4) mo_mrm_write_fluxes_states::outputdataset::idomain
```

Definition at line 46 of file mo_mrm_write_fluxes_states.f90.

18.9.3.8 nc

```
type(ncdataset) mo_mrm_write_fluxes_states::outputdataset::nc
```

Definition at line 47 of file mo_mrm_write_fluxes_states.f90.

18.9.3.9 ncdataset

```
type(ncdataset) mo_mrm_write_fluxes_states::outputdataset::ncdataset
```

Definition at line 47 of file mo_mrm_write_fluxes_states.f90.

18.9.3.10 steps

```
integer(i4) mo_mrm_write_fluxes_states::outputdataset::steps
```

Definition at line 49 of file mo_mrm_write_fluxes_states.f90.

18.9.3.11 store

```
type(outputvariable), allocatable mo_mrm_write_fluxes_states::outputdataset::store
```

Definition at line 48 of file mo_mrm_write_fluxes_states.f90.

18.9.3.12 time

```
integer(i4) mo_mrm_write_fluxes_states::outputdataset::time
```

Definition at line 49 of file mo_mrm_write_fluxes_states.f90.

18.9.3.13 to

type(ncdataset) mo_mrm_write_fluxes_states::outputdataset::to

Definition at line 47 of file mo_mrm_write_fluxes_states.f90.

18.9.3.14 variables

type(outputvariable), allocatable mo_mrm_write_fluxes_states::outputdataset::variables

Definition at line 48 of file mo_mrm_write_fluxes_states.f90.

18.9.3.15 vars

type(outputvariable), dimension(:), allocatable mo_mrm_write_fluxes_states::outputdataset↩
::vars

Definition at line 48 of file mo_mrm_write_fluxes_states.f90.

18.9.3.16 write

type(ncdataset) mo_mrm_write_fluxes_states::outputdataset::write

Definition at line 47 of file mo_mrm_write_fluxes_states.f90.

18.9.3.17 written

integer(i4) mo_mrm_write_fluxes_states::outputdataset::written

Definition at line 49 of file mo_mrm_write_fluxes_states.f90.

The documentation for this interface was generated from the following file:

- src/mRM/mo_mrm_write_fluxes_states.f90

18.10 mo_write_fluxes_states::outputdataset Interface Reference

Public Member Functions

- procedure, public updatedataset
- procedure, public writetimestep
- procedure, public close

Public Attributes

- integer(i4) idomain
- integer(i4) domain

- 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

18.10.1 Detailed Description

Definition at line 45 of file mo_write_fluxes_states.f90.

18.10.2 Member Function/Subroutine Documentation

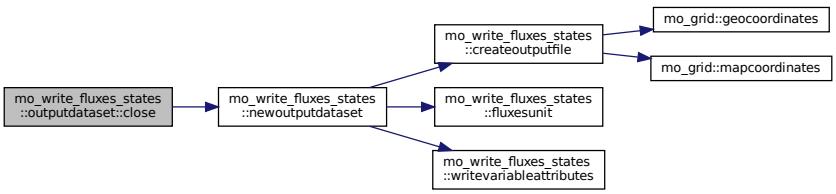
18.10.2.1 close()

`procedure, public mo_write_fluxes_states::outputdataset::close`

Definition at line 54 of file mo_write_fluxes_states.f90.

References mo_write_fluxes_states::newoutputdataset().

Here is the call graph for this function:



18.10.2.2 updatedataset()

`procedure, public mo_write_fluxes_states::outputdataset::updatedataset`

Definition at line 52 of file mo_write_fluxes_states.f90.

18.10.2.3 writetimestep()

```
procedure, public mo_write_fluxes_states::outputdataset::writetimestep
```

Definition at line 53 of file mo_write_fluxes_states.f90.

18.10.3 Member Data Documentation

18.10.3.1 all

```
type(outputvariable), allocatable mo_write_fluxes_states::outputdataset::all
```

Definition at line 48 of file mo_write_fluxes_states.f90.

18.10.3.2 count

```
integer(i4) mo_write_fluxes_states::outputdataset::count
```

Definition at line 49 of file mo_write_fluxes_states.f90.

18.10.3.3 counter

```
integer(i4) mo_write_fluxes_states::outputdataset::counter = 0
```

Definition at line 49 of file mo_write_fluxes_states.f90.

18.10.3.4 created

```
type(outputvariable), dimension (dynamic), allocatable mo_write_fluxes_states::outputdataset↩
::created
```

Definition at line 48 of file mo_write_fluxes_states.f90.

18.10.3.5 domain

```
integer(i4) mo_write_fluxes_states::outputdataset::domain
```

Definition at line 46 of file mo_write_fluxes_states.f90.

18.10.3.6 id

```
integer(i4) mo_write_fluxes_states::outputdataset::id
```

Definition at line 46 of file mo_write_fluxes_states.f90.

18.10.3.7 idomain

```
integer(i4) mo_write_fluxes_states::outputdataset::idomain
```

Definition at line 46 of file mo_write_fluxes_states.f90.

18.10.3.8 nc

```
type(ncdataset) mo_write_fluxes_states::outputdataset::nc
```

Definition at line 47 of file mo_write_fluxes_states.f90.

18.10.3.9 ncdataset

```
type(ncdataset) mo_write_fluxes_states::outputdataset::ncdataset
```

Definition at line 47 of file mo_write_fluxes_states.f90.

18.10.3.10 steps

```
integer(i4) mo_write_fluxes_states::outputdataset::steps
```

Definition at line 49 of file mo_write_fluxes_states.f90.

18.10.3.11 store

```
type(outputvariable), allocatable mo_write_fluxes_states::outputdataset::store
```

Definition at line 48 of file mo_write_fluxes_states.f90.

18.10.3.12 time

```
integer(i4) mo_write_fluxes_states::outputdataset::time
```

Definition at line 49 of file mo_write_fluxes_states.f90.

18.10.3.13 to

```
type(ncdataset) mo_write_fluxes_states::outputdataset::to
```

Definition at line 47 of file mo_write_fluxes_states.f90.

18.10.3.14 variables

```
type(outputvariable), allocatable mo_write_fluxes_states::outputdataset::variables
```

Definition at line 48 of file mo_write_fluxes_states.f90.

18.10.3.15 vars

```
type(outputvariable), dimension(:), allocatable mo_write_fluxes_states::outputdataset::vars
```

Definition at line 48 of file mo_write_fluxes_states.f90.

18.10.3.16 write

```
type(ncdataset) mo_write_fluxes_states::outputdataset::write
```

Definition at line 47 of file mo_write_fluxes_states.f90.

18.10.3.17 written

```
integer(i4) mo_write_fluxes_states::outputdataset::written
```

Definition at line 49 of file mo_write_fluxes_states.f90.

The documentation for this interface was generated from the following file:

- src/mHM/mo_write_fluxes_states.f90

18.11 mo_mrm_write_fluxes_states::outputvariable Interface Reference

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

18.11.1 Detailed Description

Definition at line 27 of file mo_mrm_write_fluxes_states.f90.

18.11.2 Member Function/Subroutine Documentation

18.11.2.1 updatevariable()

```
procedure, public mo_mrm_write_fluxes_states::outputvariable::updatevariable
```

Definition at line 35 of file mo_mrm_write_fluxes_states.f90.

18.11.2.2 writevariabletimestep()

```
procedure, public mo_mrm_write_fluxes_states::outputvariable::writevariabletimestep
```

Definition at line 36 of file mo_mrm_write_fluxes_states.f90.

18.11.3 Member Data Documentation

18.11.3.1 average

```
logical mo_mrm_write_fluxes_states::outputvariable::average
```

Definition at line 29 of file mo_mrm_write_fluxes_states.f90.

18.11.3.2 avg

```
logical mo_mrm_write_fluxes_states::outputvariable::avg = .false.
```

Definition at line 29 of file mo_mrm_write_fluxes_states.f90.

18.11.3.3 before

```
logical mo_mrm_write_fluxes_states::outputvariable::before
```

Definition at line 29 of file mo_mrm_write_fluxes_states.f90.

18.11.3.4 between

```
real(dp), allocatable mo_mrm_write_fluxes_states::outputvariable::between
```

Definition at line 31 of file mo_mrm_write_fluxes_states.f90.

18.11.3.5 calls

```
integer(i4) mo_mrm_write_fluxes_states::outputvariable::calls
```

Definition at line 32 of file mo_mrm_write_fluxes_states.f90.

18.11.3.6 contains

```
type(ncvariable) mo_mrm_write_fluxes_states::outputvariable::contains
```

Definition at line 28 of file mo_mrm_write_fluxes_states.f90.

18.11.3.7 count

```
integer(i4) mo_mrm_write_fluxes_states::outputvariable::count
```

Definition at line 32 of file mo_mrm_write_fluxes_states.f90.

18.11.3.8 counter

```
integer(i4) mo_mrm_write_fluxes_states::outputvariable::counter = 0
```

Definition at line 32 of file mo_mrm_write_fluxes_states.f90.

18.11.3.9 data [1/2]

```
logical, dimension(:), allocatable, pointer mo_mrm_write_fluxes_states::outputvariable::data
```

Definition at line 29 of file mo_mrm_write_fluxes_states.f90.

18.11.3.10 data [2/2]

```
real(dp), dimension(:), allocatable, pointer mo_mrm_write_fluxes_states::outputvariable::data
```

Definition at line 31 of file mo_mrm_write_fluxes_states.f90.

18.11.3.11 mask

```
logical, dimension(:, :), pointer mo_mrm_write_fluxes_states::outputvariable::mask
```

Definition at line 30 of file mo_mrm_write_fluxes_states.f90.

18.11.3.12 nc

```
type(ncvariable) mo_mrm_write_fluxes_states::outputvariable::nc
```

Definition at line 28 of file mo_mrm_write_fluxes_states.f90.

18.11.3.13 ncdataset

```
type(ncvariable) mo_mrm_write_fluxes_states::outputvariable::ncdataset
```

Definition at line 28 of file mo_mrm_write_fluxes_states.f90.

18.11.3.14 number

```
integer(i4) mo_mrm_write_fluxes_states::outputvariable::number
```

Definition at line 32 of file mo_mrm_write_fluxes_states.f90.

18.11.3.15 of

```
integer(i4) mo_mrm_write_fluxes_states::outputvariable::of
```

Definition at line 32 of file mo_mrm_write_fluxes_states.f90.

18.11.3.16 reconstruct

```
logical, pointer mo_mrm_write_fluxes_states::outputvariable::reconstruct
```

Definition at line 30 of file mo_mrm_write_fluxes_states.f90.

18.11.3.17 store

```
real(dp), allocatable mo_mrm_write_fluxes_states::outputvariable::store
```

Definition at line 31 of file mo_mrm_write_fluxes_states.f90.

18.11.3.18 the [1/3]

```
type(ncvariable), allocatable mo_mrm_write_fluxes_states::outputvariable::the
```

Definition at line 28 of file mo_mrm_write_fluxes_states.f90.

18.11.3.19 the [2/3]

```
real(dp), allocatable mo_mrm_write_fluxes_states::outputvariable::the
```

Definition at line 31 of file mo_mrm_write_fluxes_states.f90.

18.11.3.20 the [3/3]

```
integer(i4), allocatable mo_mrm_write_fluxes_states::outputvariable::the
```

Definition at line 32 of file mo_mrm_write_fluxes_states.f90.

18.11.3.21 to

```
logical, pointer mo_mrm_write_fluxes_states::outputvariable::to
```

Definition at line 30 of file mo_mrm_write_fluxes_states.f90.

18.11.3.22 updatevariable

```
integer(i4), public mo_mrm_write_fluxes_states::outputvariable::updatevariable
```

Definition at line 32 of file mo_mrm_write_fluxes_states.f90.

18.11.3.23 variable

```
type(ncvariable) mo_mrm_write_fluxes_states::outputvariable::variable
```

Definition at line 28 of file mo_mrm_write_fluxes_states.f90.

18.11.3.24 which

```
type(ncvariable) mo_mrm_write_fluxes_states::outputvariable::which
```

Definition at line 28 of file mo_mrm_write_fluxes_states.f90.

18.11.3.25 writes

```
real(dp), allocatable mo_mrm_write_fluxes_states::outputvariable::writes
```

Definition at line 31 of file mo_mrm_write_fluxes_states.f90.

18.11.3.26 writing

```
logical mo_mrm_write_fluxes_states::outputvariable::writing
```

Definition at line 29 of file mo_mrm_write_fluxes_states.f90.

The documentation for this interface was generated from the following file:

- src/mRM/mo_mrm_write_fluxes_states.f90

18.12 mo_write_fluxes_states::outputvariable Interface Reference

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

18.12.1 Detailed Description

Definition at line 27 of file mo_write_fluxes_states.f90.

18.12.2 Member Function/Subroutine Documentation

18.12.2.1 updatevariable()

```
procedure, public mo_write_fluxes_states::outputvariable::updatevariable
```

Definition at line 35 of file mo_write_fluxes_states.f90.

18.12.2.2 writevariabletimestep()

```
procedure, public mo_write_fluxes_states::outputvariable::writevariabletimestep
```

Definition at line 36 of file mo_write_fluxes_states.f90.

18.12.3 Member Data Documentation

18.12.3.1 average

```
logical mo_write_fluxes_states::outputvariable::average
```

Definition at line 29 of file mo_write_fluxes_states.f90.

18.12.3.2 avg

```
logical mo_write_fluxes_states::outputvariable::avg = .false.
```

Definition at line 29 of file mo_write_fluxes_states.f90.

18.12.3.3 before

```
logical mo_write_fluxes_states::outputvariable::before
```

Definition at line 29 of file mo_write_fluxes_states.f90.

18.12.3.4 between

```
real(dp), allocatable mo_write_fluxes_states::outputvariable::between
```

Definition at line 31 of file mo_write_fluxes_states.f90.

18.12.3.5 calls

```
integer(i4) mo_write_fluxes_states::outputvariable::calls
```

Definition at line 32 of file mo_write_fluxes_states.f90.

18.12.3.6 contains

```
type(ncvariable) mo_write_fluxes_states::outputvariable::contains
```

Definition at line 28 of file mo_write_fluxes_states.f90.

18.12.3.7 count

```
integer(i4) mo_write_fluxes_states::outputvariable::count
```

Definition at line 32 of file mo_write_fluxes_states.f90.

18.12.3.8 counter

```
integer(i4) mo_write_fluxes_states::outputvariable::counter = 0
```

Definition at line 32 of file mo_write_fluxes_states.f90.

18.12.3.9 data [1/2]

```
logical, dimension(:), allocatable, pointer mo_write_fluxes_states::outputvariable::data
```

Definition at line 29 of file mo_write_fluxes_states.f90.

18.12.3.10 data [2/2]

```
real(dp), dimension(:), allocatable, pointer mo_write_fluxes_states::outputvariable::data
```

Definition at line 31 of file mo_write_fluxes_states.f90.

18.12.3.11 mask

```
logical, dimension(:, :), pointer mo_write_fluxes_states::outputvariable::mask
```

Definition at line 30 of file mo_write_fluxes_states.f90.

18.12.3.12 nc

```
type(ncvariable) mo_write_fluxes_states::outputvariable::nc
```

Definition at line 28 of file mo_write_fluxes_states.f90.

18.12.3.13 ncdataset

```
type(ncvariable) mo_write_fluxes_states::outputvariable::ncdataset
```

Definition at line 28 of file mo_write_fluxes_states.f90.

18.12.3.14 number

```
integer(i4) mo_write_fluxes_states::outputvariable::number
```

Definition at line 32 of file mo_write_fluxes_states.f90.

18.12.3.15 of

```
integer(i4) mo_write_fluxes_states::outputvariable::of
```

Definition at line 32 of file mo_write_fluxes_states.f90.

18.12.3.16 reconstruct

```
logical, pointer mo_write_fluxes_states::outputvariable::reconstruct
```

Definition at line 30 of file mo_write_fluxes_states.f90.

18.12.3.17 store

```
real(dp), allocatable mo_write_fluxes_states::outputvariable::store
```

Definition at line 31 of file mo_write_fluxes_states.f90.

18.12.3.18 the [1/3]

```
type(ncvariable), allocatable mo_write_fluxes_states::outputvariable::the
```

Definition at line 28 of file mo_write_fluxes_states.f90.

18.12.3.19 the [2/3]

```
real(dp), allocatable mo_write_fluxes_states::outputvariable::the
```

Definition at line 31 of file mo_write_fluxes_states.f90.

18.12.3.20 the [3/3]

```
integer(i4), allocatable mo_write_fluxes_states::outputvariable::the
```

Definition at line 32 of file mo_write_fluxes_states.f90.

18.12.3.21 to

```
logical, pointer mo_write_fluxes_states::outputvariable::to
```

Definition at line 30 of file mo_write_fluxes_states.f90.

18.12.3.22 updatevariable

```
integer(i4), public mo_write_fluxes_states::outputvariable::updatevariable
```

Definition at line 32 of file mo_write_fluxes_states.f90.

18.12.3.23 variable

```
type(ncvariable) mo_write_fluxes_states::outputvariable::variable
```

Definition at line 28 of file mo_write_fluxes_states.f90.

18.12.3.24 which

```
type(ncvariable) mo_write_fluxes_states::outputvariable::which
```

Definition at line 28 of file mo_write_fluxes_states.f90.

18.12.3.25 writes

```
real(dp), allocatable mo_write_fluxes_states::outputvariable::writes
```

Definition at line 31 of file mo_write_fluxes_states.f90.

18.12.3.26 writing

```
logical mo_write_fluxes_states::outputvariable::writing
```

Definition at line 29 of file mo_write_fluxes_states.f90.

The documentation for this interface was generated from the following file:

- src/mHM/mo_write_fluxes_states.f90

18.13 mo_common_variables::period Type Reference

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

18.13.1 Detailed Description

Definition at line 61 of file mo_common_variables.f90.

18.13.2 Member Data Documentation

18.13.2.1 dend

```
integer(i4) mo_common_variables::period::dend
```

Definition at line 65 of file mo_common_variables.f90.

18.13.2.2 dstart

```
integer(i4) mo_common_variables::period::dstart
```

Definition at line 62 of file mo_common_variables.f90.

18.13.2.3 julend

```
integer(i4) mo_common_variables::period::julend
```

Definition at line 69 of file mo_common_variables.f90.

18.13.2.4 julstart

```
integer(i4) mo_common_variables::period::julstart
```

Definition at line 68 of file mo_common_variables.f90.

18.13.2.5 mend

```
integer(i4) mo_common_variables::period::mend
```

Definition at line 66 of file mo_common_variables.f90.

18.13.2.6 mstart

```
integer(i4) mo_common_variables::period::mstart
```

Definition at line 63 of file mo_common_variables.f90.

18.13.2.7 nobs

```
integer(i4) mo_common_variables::period::nobs
```

Definition at line 70 of file mo_common_variables.f90.

18.13.2.8 yend

```
integer(i4) mo_common_variables::period::yend
```

Definition at line 67 of file mo_common_variables.f90.

18.13.2.9 ystart

```
integer(i4) mo_common_variables::period::ystart
```

Definition at line 64 of file mo_common_variables.f90.

The documentation for this type was generated from the following file:

- src/common/mo_common_variables.f90

18.14 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.

18.14.1 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

Definition at line 52 of file mo_read_spatial_data.f90.

18.14.2 Member Function/Subroutine Documentation

18.14.2.1 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

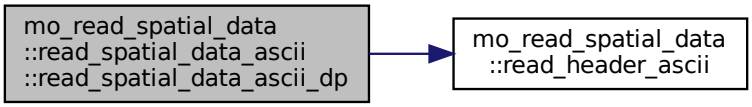
Date

Jun 2018

Definition at line 94 of file mo_read_spatial_data.f90.

References mo_read_spatial_data::read_header_ascii().

Here is the call graph for this function:



18.14.2.2 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

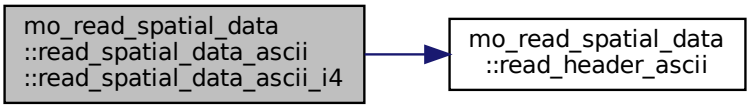
Date

Jun 2018

Definition at line 234 of file mo_read_spatial_data.f90.

References mo_read_spatial_data::read_header_ascii().

Here is the call graph for this function:



The documentation for this interface was generated from the following file:

- src/common/mo_read_spatial_data.f90

18.15 mo_mrm_riv_temp_class::riv_temp_type Module Reference

This is a container to define the river temperature routing in the current time step.

Public Member Functions

- procedure config
  configure the riv_temp_type class from the mhm namelist
- procedure init
  initalize the riv_temp_type class for the current domain
- procedure init_area
  initialize the river area of riv_temp_type class for the current domain
- procedure init_riv_temp
  initialize the river temperature of riv_temp_type class for the current domain
- procedure acc_source_e
  accumulate energy sources of riv_temp_type
- procedure finalize_source_e
  finalize energy sources of riv_temp_type
- procedure get_lrad_out
  get outgoing longwave radiation of riv_temp_type
- procedure get_lat_heat
  latent heat flux of riv_temp_type
- procedure get_sens_heat
  sensible heat flux of riv_temp_type
- procedure get_e_io
  get complete energy source of riv_temp_type at given cell
- procedure l11_routing_e
  execute the temperature routing of riv_temp_type
- procedure init_iter
  initialize iterative solver of riv_temp_type
- procedure next_iter
  execute next iteration with iterative solver of riv_temp_type
- procedure reset_timestep
  reset riv_temp_type class for next timestep

- procedure alloc_lateral

  allocate lateral temp components of riv_temp_type class for current domain
- procedure dealloc_lateral

  deallocate lateral temp components of riv_temp_type

Public Attributes

- logical active = .false.

  state if this process is active
- integer(i4) case = 0_i4

  the selected process-case option
- character(256) nml_name = 'config_riv_temp'

  namelist name in mhm namelist
- character(256), dimension(:), allocatable dir_riv_widths

  Directory where river widths are stored.
- character(256) riv_widths_file

  file name for river widths
- character(256) riv_widths_name

  variable name for river widths
- real(dp), dimension(:), allocatable, public l11_riv_widths

  river widths in L11
- real(dp), dimension(:), allocatable, public l11_riv_areas

  river area in L11
- real(dp) albedo_water

  albedo of open water
- real(dp) pt_a_water

  priestley taylor alpha parameter for PET on open water
- real(dp) emissivity_water

  emissivity of water
- real(dp) turb_heat_ex_coeff

  lateral heat exchange coefficient water <-> air
- real(dp) delta_t = 0.1_dp

  cutoff value for temperature
- integer(i4) max_iter

  input: maximal number of iterations done
- real(dp) delta_iter

  input: convergence criteria for iterative solver
- real(dp) step_iter

  input: step-size for linear search
- logical first_iter

  whether it is at the first iteration (to determine search direction)
- logical up_iter

  whether the search direction is upwards
- logical bisect_iter

  whether to do the bisection search part (after the interval is found)
- real(dp) up_bnd_iter

  upper bound for the current bisection step
- real(dp) low_bnd_iter

  lower bound for the current bisection step
- real(dp), dimension(:), allocatable l1_runoff_e

runoff energy at L1 level

- real(dp), dimension(:), allocatable l1_acc_ssrd

  accumulated shortwave radiation at L1 level
- real(dp), dimension(:), allocatable l1_acc_strd

  accumulated longwave radiation at L1 level
- real(dp), dimension(:), allocatable l1_acc_temp

  accumulated air temperature radiation at L1 level
- integer(i4) ts_cnt

  sub time-step counter for accumulation of meteo
- integer(i4) s11

  starting index for current L11 domain
- integer(i4) e11

  ending index for current L11 domain
- real(dp), dimension(:,:), allocatable netnode_e_in

  Total energy inputs at t-1 and t.
- real(dp), dimension(:,:), allocatable netnode_e_r

  energy leaving at t-1 and t
- real(dp), dimension(:), allocatable netnode_e_mod

  Simulated routed energy.
- real(dp), dimension(:), allocatable netnode_e_out

  total energy source from cell in L11
- real(dp), dimension(:), allocatable l11_srad_net

  net short wave radiation at L11
- real(dp), dimension(:), allocatable l11_lrad_in

  incoming long wave radiation at L11
- real(dp), dimension(:), allocatable l11_air_temp

  air temp at L11
- real(dp), dimension(:), allocatable river_temp

  resulting river temp at L11 in [deg C]

18.15.1 Detailed Description

This is a container to define the river temperature routing in the current time step.

This class provides all procedures to rout river tmperature through the river network.

Warning

> This feature is still experimental! This first version doesn't provide ice covering, which means, that the river temperature can drop below 0 [deg C] in winter.

Definition at line 23 of file mo_mrm_riv_temp_class.f90.

18.15.2 Member Function/Subroutine Documentation

18.15.2.1 acc_source_e()

```
procedure mo_mrm_riv_temp_class::riv_temp_type::acc_source_e
```

accumulate energy sources of riv_temp_type

Parameters

in	*time*	current decimal Julian day
in	*ntimesteps_day*	number of time intervals per day, transformed in dp
in	*fsealed_area_fraction*	sealed area fraction [1]
in	*fast_interflow*	q_0 Fast runoff component [mm TS-1]
in	*slow_interflow*	q_1 Slow runoff component [mm TS-1]
in	*baseflow*	q_2 Baseflow [mm TS-1]
in	*direct_runoff*	q_D Direct runoff from impervious areas [mm TS-1]
in	*temp_air*	air temperature [K]
in	*mean_temp_air*	annual mean air temperature [K]
in	*ssrd_day*	Daily mean short radiation
in	*strd_day*	Daily mean longwave radiation
in	*read_meteo_weights*	flag whether weights for tavg and pet have read and should be used
in	*temp_weights*	multiplicative weights for temperature (deg K)
in	*fday_temp*	[-] day factor mean temp
in	*fnight_temp*	[-] night factor mean temp
in	*fday_ssrd*	Daytime fraction of ssrd
in	*fnight_ssrd*	Nighttime fraction of ssrd
in	*fday_strd*	Daytime fraction of strd
in	*fnight_strd*	Nighttime fraction of strd

See also

mo_mrm_riv_temp_class::acc_source_e

Definition at line 82 of file mo_mrm_riv_temp_class.f90.

18.15.2.2 alloc_lateral()

`procedure mo_mrm_riv_temp_class::riv_temp_type::alloc_lateral`

allocate lateral temp components of riv_temp_type class for current domain

Parameters

in	*ncells*	number of cells for the current domain

See also

mo_mrm_riv_temp_class::alloc_lateral

Definition at line 108 of file mo_mrm_riv_temp_class.f90.

18.15.2.3 config()

`procedure mo_mrm_riv_temp_class::riv_temp_type::config`

configure the riv_temp_type class from the mhm namelist

Parameters

in	*file_namelist*	mhm namelist file
in	*file_namelist_param*	mhm parameter namelist file

See also

mo_mrm_riv_temp_class::config

Definition at line 72 of file mo_mrm_riv_temp_class.f90.

18.15.2.4 dealloc_lateral()

`procedure mo_mrm_riv_temp_class::riv_temp_type::dealloc_lateral`

deallocate lateral temp components of riv_temp_type

See also

mo_mrm_riv_temp_class::dealloc_lateral

Definition at line 110 of file mo_mrm_riv_temp_class.f90.

18.15.2.5 finalize_source_e()

`procedure mo_mrm_riv_temp_class::riv_temp_type::finalize_source_e`

finalize energy sources of riv_temp_type

Parameters

in	*efecarea*	effective area in [km2] at Level 1
in	*l1_l11_id*	L11 Ids mapped on L1
in	*l11_areacell*	effective area in [km2] at Level 11
in	*l11_l1_id*	L1 Ids mapped on L11
in	*timestep*	simulation timestep in [h]
in	*map_flag*	Flag indicating whether routing resolution is higher than hydrologic one

See also

mo_mrm_riv_temp_class::finalize_source_e

Definition at line 84 of file mo_mrm_riv_temp_class.f90.

18.15.2.6 get_e_io()

`procedure mo_mrm_riv_temp_class::riv_temp_type::get_e_io`

get complete energy source of riv_temp_type at given cell

Returns

energy IO

Parameters

in	*riv_temp*	given river temperature in K to calculate heat fluxes
in	*cell*	cell index in the current domain

See also

mo_mrm_riv_temp_class::get_e_io

Definition at line 94 of file mo_mrm_riv_temp_class.f90.

18.15.2.7 get_lat_heat()

`procedure mo_mrm_riv_temp_class::riv_temp_type::get_lat_heat`

latent heat flux of riv_temp_type

Returns

latent heat flux

Parameters

in	*air_temp*	air temperature in deg C
in	*netrad*	net radiation in W $*$ m-2

See also

mo_mrm_riv_temp_class::get_lat_heat

Definition at line 90 of file mo_mrm_riv_temp_class.f90.

18.15.2.8 get_lrad_out()

`procedure mo_mrm_riv_temp_class::riv_temp_type::get_lrad_out`

get outgoing longwave radiation of riv_temp_type

Returns

outgoing longwave radiation

Parameters

in	*riv_temp*	river temperature in K

See also

mo_mrm_riv_temp_class::get_lrad_out

Definition at line 88 of file mo_mrm_riv_temp_class.f90.

18.15.2.9 get_sens_heat()

`procedure mo_mrm_riv_temp_class::riv_temp_type::get_sens_heat`

sensible heat flux of riv_temp_type

Returns

sensible heat flux

Parameters

in	*air_temp*	air temperature in [deg C]
in	*riv_temp*	river temperature in [deg C]

See also

mo_mrm_riv_temp_class::get_sens_heat

Definition at line 92 of file mo_mrm_riv_temp_class.f90.

18.15.2.10 init()

`procedure mo_mrm_riv_temp_class::riv_temp_type::init`

initalize the riv_temp_type class for the current domain

Parameters

in	*ncells*	number of cells for the current domain

See also

mo_mrm_riv_temp_class::init

Definition at line 74 of file mo_mrm_riv_temp_class.f90.

18.15.2.11 init_area()

`procedure mo_mrm_riv_temp_class::riv_temp_type::init_area`

initialize the river area of riv_temp_type class for the current domain

Parameters

in	*idomain*	Domain ID

Parameters

in	*l11_netperm*	L11 routing order
in	*l11_fromn*	L11 source grid cell order
in	*l11_length*	L11 link length
in	*nlinks*	number of L11 links in the current domain
in	*ncells*	number of L11 cells of the current domain
in	*ncols*	Number of columns
in	*nrows*	Number of rows
in	*l11_mask*	the mask for valid cells in the original grid (nrows, ncols)

See also

mo_mrm_riv_temp_class::init_area

Definition at line 76 of file mo_mrm_riv_temp_class.f90.

18.15.2.12 init_iter()

`procedure mo_mrm_riv_temp_class::riv_temp_type::init_iter`

initialize iterative solver of riv_temp_type

See also

mo_mrm_riv_temp_class::init_iter

Definition at line 100 of file mo_mrm_riv_temp_class.f90.

18.15.2.13 init_riv_temp()

`procedure mo_mrm_riv_temp_class::riv_temp_type::init_riv_temp`

initialize the river temperature of riv_temp_type class for the current domain

Parameters

in	*time*	current decimal Julian day
in	*ntimesteps_day*	number of time intervals per day, transformed in dp
in	*temp_air*	air temperature [K]
in	*read_meteo_weights*	flag whether weights for tavg and pet have read and should be used
in	*temp_weights*	multiplicative weights for temperature (deg K)
in	*fday_temp*	[-] day factor mean temp
in	*fnight_temp*	[-] night factor mean temp
in	*efecarea*	effective area in [km2] at Level 1
in	*l1_l11_id*	L11 Ids mapped on L1
in	*l11_areacell*	effective area in [km2] at Level 11
in	*l11_l1_id*	L1 Ids mapped on L11
in	*map_flag*	Flag indicating whether routing resolution is higher than hydrologic one

See also

mo_mrm_riv_temp_class::init_riv_temp

Definition at line 78 of file mo_mrm_riv_temp_class.f90.

18.15.2.14 l11_routing_e()

`procedure mo_mrm_riv_temp_class::riv_temp_type::l11_routing_e`

execute the temperature routing of riv_temp_type

Parameters

in	*nlinks*	number of stream segment (reaches)
in	*netperm*	routing order of a given domain (permutation)
in	*netlink_fromn*	from node
in	*netlink_ton*	to node
in	*netlink_c1*	routing parameter C1 ([7] p. 25-41)
in	*netlink_c2*	routing parameters C2 (id)
in	*ninflowgauges*	[-] number of inflow points
in	*inflowheadwater*	[-] if to consider headwater cells of inflow gauge
in	*inflownodelist*	[-] L11 ID of inflow points
in	*l11_qtr*	[m3 s-1] Transformed outflow leaving node I at current timestep(Muskingum)
in	*l11_qmod*	[m3 s-1] Simulated routed discharge

See also

mo_mrm_riv_temp_class::l11_routing_e

Definition at line 96 of file mo_mrm_riv_temp_class.f90.

18.15.2.15 next_iter()

`procedure mo_mrm_riv_temp_class::riv_temp_type::next_iter`

execute next iteration with iterative solver of riv_temp_type

Parameters

in,out	*t_est*	estimated river temperature
in	*t_rout*	calculated (routed) river temperature

See also

mo_mrm_riv_temp_class::next_iter

Definition at line 102 of file mo_mrm_riv_temp_class.f90.

18.15.2.16 reset_timestep()

`procedure mo_mrm_riv_temp_class::riv_temp_type::reset_timestep`

reset riv_temp_type class for next timestep

See also

mo_mrm_riv_temp_class::reset_timestep

Definition at line 106 of file mo_mrm_riv_temp_class.f90.

18.15.3 Member Data Documentation

18.15.3.1 active

`logical mo_mrm_riv_temp_class::riv_temp_type::active = .false.`

state if this process is active

Definition at line 25 of file mo_mrm_riv_temp_class.f90.

18.15.3.2 albedo_water

`real(dp) mo_mrm_riv_temp_class::riv_temp_type::albedo_water`

albedo of open water

Definition at line 35 of file mo_mrm_riv_temp_class.f90.

18.15.3.3 bisect_iter

`logical mo_mrm_riv_temp_class::riv_temp_type::bisect_iter`

whether to do the bisection search part (after the interval is found)

Definition at line 48 of file mo_mrm_riv_temp_class.f90.

18.15.3.4 case

`integer(i4) mo_mrm_riv_temp_class::riv_temp_type::case = 0_i4`

the selected process-case option

Definition at line 26 of file mo_mrm_riv_temp_class.f90.

18.15.3.5 delta_iter

`real(dp) mo_mrm_riv_temp_class::riv_temp_type::delta_iter`

input: convergence criteria for iterative solver

Definition at line 44 of file mo_mrm_riv_temp_class.f90.

18.15.3.6 delta_t

```
real(dp) mo_mrm_riv_temp_class::riv_temp_type::delta_t = 0.1_dp
```

cutoff value for temperature

Definition at line 41 of file mo_mrm_riv_temp_class.f90.

18.15.3.7 dir_riv_widths

```
character(256), dimension(:), allocatable mo_mrm_riv_temp_class::riv_temp_type::dir_riv_widths
```

Directory where river widths are stored.

Definition at line 29 of file mo_mrm_riv_temp_class.f90.

18.15.3.8 e11

```
integer(i4) mo_mrm_riv_temp_class::riv_temp_type::e11
```

ending index for current L11 domain

Definition at line 59 of file mo_mrm_riv_temp_class.f90.

18.15.3.9 emissivity_water

```
real(dp) mo_mrm_riv_temp_class::riv_temp_type::emissivity_water
```

emissivity of water

Definition at line 38 of file mo_mrm_riv_temp_class.f90.

18.15.3.10 first_iter

```
logical mo_mrm_riv_temp_class::riv_temp_type::first_iter
```

whether it is at the first iteration (to determine search direction)

Definition at line 46 of file mo_mrm_riv_temp_class.f90.

18.15.3.11 l11_air_temp

```
real(dp), dimension(:), allocatable mo_mrm_riv_temp_class::riv_temp_type::l11_air_temp
```

air temp at L11

Definition at line 66 of file mo_mrm_riv_temp_class.f90.

18.15.3.12 l11_lrad_in

```
real(dp), dimension(:), allocatable mo_mrm_riv_temp_class::riv_temp_type::l11_lrad_in
```

incoming long wave radiation at L11

Definition at line 65 of file mo_mrm_riv_temp_class.f90.

18.15.3.13 l11_riv_areas

```
real(dp), dimension(:), allocatable, public mo_mrm_riv_temp_class::riv_temp_type::l11_riv_↩
areas
```

river area in L11

Definition at line 33 of file mo_mrm_riv_temp_class.f90.

18.15.3.14 l11_riv_widths

```
real(dp), dimension(:), allocatable, public mo_mrm_riv_temp_class::riv_temp_type::l11_riv_↩
widths
```

river widths in L11

Definition at line 32 of file mo_mrm_riv_temp_class.f90.

18.15.3.15 l11_srad_net

```
real(dp), dimension(:), allocatable mo_mrm_riv_temp_class::riv_temp_type::l11_srad_net
```

net short wave radiation at L11

Definition at line 64 of file mo_mrm_riv_temp_class.f90.

18.15.3.16 l1_acc_ssrd

```
real(dp), dimension(:), allocatable mo_mrm_riv_temp_class::riv_temp_type::l1_acc_ssrd
```

accumulated shortwave radiation at L1 level

Definition at line 53 of file mo_mrm_riv_temp_class.f90.

18.15.3.17 l1_acc_strd

```
real(dp), dimension(:), allocatable mo_mrm_riv_temp_class::riv_temp_type::l1_acc_strd
```

accumulated longwave radiation at L1 level

Definition at line 54 of file mo_mrm_riv_temp_class.f90.

18.15.3.18 l1_acc_temp

```
real(dp), dimension(:), allocatable mo_mrm_riv_temp_class::riv_temp_type::l1_acc_temp
```

accumulated air temperature radiation at L1 level

Definition at line 55 of file mo_mrm_riv_temp_class.f90.

18.15.3.19 l1_runoff_e

```
real(dp), dimension(:), allocatable mo_mrm_riv_temp_class::riv_temp_type::l1_runoff_e
```

runoff energy at L1 level

Definition at line 52 of file mo_mrm_riv_temp_class.f90.

18.15.3.20 low_bnd_iter

```
real(dp) mo_mrm_riv_temp_class::riv_temp_type::low_bnd_iter
```

lower bound for the current bisection step

Definition at line 50 of file mo_mrm_riv_temp_class.f90.

18.15.3.21 max_iter

```
integer(i4) mo_mrm_riv_temp_class::riv_temp_type::max_iter
```

input: maximal number of iterations done

Definition at line 43 of file mo_mrm_riv_temp_class.f90.

18.15.3.22 netnode_e_in

```
real(dp), dimension(:,:), allocatable mo_mrm_riv_temp_class::riv_temp_type::netnode_e_in
```

Total energy inputs at t-1 and t.

Definition at line 60 of file mo_mrm_riv_temp_class.f90.

18.15.3.23 netnode_e_mod

```
real(dp), dimension(:), allocatable mo_mrm_riv_temp_class::riv_temp_type::netnode_e_mod
```

Simulated routed energy.

Definition at line 62 of file mo_mrm_riv_temp_class.f90.

18.15.3.24 netnode_e_out

```
real(dp), dimension(:), allocatable mo_mrm_riv_temp_class::riv_temp_type::netnode_e_out
```

total energy source from cell in L11

Definition at line 63 of file mo_mrm_riv_temp_class.f90.

18.15.3.25 netnode_e_r

```
real(dp), dimension(:,:), allocatable mo_mrm_riv_temp_class::riv_temp_type::netnode_e_r
```

energy leaving at t-1 and t

Definition at line 61 of file mo_mrm_riv_temp_class.f90.

18.15.3.26 nml_name

```
character(256) mo_mrm_riv_temp_class::riv_temp_type::nml_name = 'config_riv_temp'
```

namelist name in mhm namelist

Definition at line 27 of file mo_mrm_riv_temp_class.f90.

18.15.3.27 pt_a_water

```
real(dp) mo_mrm_riv_temp_class::riv_temp_type::pt_a_water
```

priestley taylor alpha parameter for PET on open water

Definition at line 36 of file mo_mrm_riv_temp_class.f90.

18.15.3.28 riv_widths_file

```
character(256) mo_mrm_riv_temp_class::riv_temp_type::riv_widths_file
```

file name for river widths

Definition at line 30 of file mo_mrm_riv_temp_class.f90.

18.15.3.29 riv_widths_name

```
character(256) mo_mrm_riv_temp_class::riv_temp_type::riv_widths_name
```

variable name for river widths

Definition at line 31 of file mo_mrm_riv_temp_class.f90.

18.15.3.30 river_temp

```
real(dp), dimension(:), allocatable mo_mrm_riv_temp_class::riv_temp_type::river_temp
```

resulting river temp at L11 in [deg C]

Definition at line 68 of file mo_mrm_riv_temp_class.f90.

18.15.3.31 s11

`integer(i4) mo_mrm_riv_temp_class::riv_temp_type::s11`

starting index for current L11 domain

Definition at line 58 of file mo_mrm_riv_temp_class.f90.

18.15.3.32 step_iter

`real(dp) mo_mrm_riv_temp_class::riv_temp_type::step_iter`

input: step-size for linear search

Definition at line 45 of file mo_mrm_riv_temp_class.f90.

18.15.3.33 ts_cnt

`integer(i4) mo_mrm_riv_temp_class::riv_temp_type::ts_cnt`

sub time-step counter for accumulation of meteo

Definition at line 56 of file mo_mrm_riv_temp_class.f90.

18.15.3.34 turb_heat_ex_coeff

`real(dp) mo_mrm_riv_temp_class::riv_temp_type::turb_heat_ex_coeff`

lateral heat exchange coefficient water <-> air

Definition at line 39 of file mo_mrm_riv_temp_class.f90.

18.15.3.35 up_bnd_iter

`real(dp) mo_mrm_riv_temp_class::riv_temp_type::up_bnd_iter`

upper bound for the current bisection step

Definition at line 49 of file mo_mrm_riv_temp_class.f90.

18.15.3.36 up_iter

`logical mo_mrm_riv_temp_class::riv_temp_type::up_iter`

whether the search direction is upwards

Definition at line 47 of file mo_mrm_riv_temp_class.f90.

The documentation for this module was generated from the following file:

- src/mRM/mo_mrm_riv_temp_class.f90

18.16 mo_mpr_global_variables::soiltype Type Reference

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

18.16.1 Detailed Description

Definition at line 34 of file mo_mpr_global_variables.f90.

18.16.2 Member Data Documentation

18.16.2.1 clay

```
real(dp), dimension(:, :), allocatable mo_mpr_global_variables::soiltype::clay  [private]
```

Definition at line 45 of file mo_mpr_global_variables.f90.

18.16.2.2 db

```
real(dp), dimension(:, :, :), allocatable mo_mpr_global_variables::soiltype::db  [private]
```

Definition at line 59 of file mo_mpr_global_variables.f90.

18.16.2.3 dbm

```
real(dp), dimension(:, :), allocatable mo_mpr_global_variables::soiltype::dbm  [private]
```

Definition at line 47 of file mo_mpr_global_variables.f90.

18.16.2.4 depth

```
real(dp), dimension(:, :), allocatable mo_mpr_global_variables::soiltype::depth [private]
```

Definition at line 48 of file mo_mpr_global_variables.f90.

18.16.2.5 id

```
integer(i4), dimension(:), allocatable mo_mpr_global_variables::soiltype::id [private]
```

Definition at line 39 of file mo_mpr_global_variables.f90.

18.16.2.6 is_present

```
integer(i4), dimension(:), allocatable mo_mpr_global_variables::soiltype::is_present [private]
```

Definition at line 41 of file mo_mpr_global_variables.f90.

18.16.2.7 ks

```
real(dp), dimension(:, :, :), allocatable mo_mpr_global_variables::soiltype::ks [private]
```

Definition at line 67 of file mo_mpr_global_variables.f90.

18.16.2.8 ld

```
real(dp), dimension(:, :), allocatable mo_mpr_global_variables::soiltype::ld [private]
```

Definition at line 44 of file mo_mpr_global_variables.f90.

18.16.2.9 nhorizons

```
integer(i4), dimension(:), allocatable mo_mpr_global_variables::soiltype::nhorizons [private]
```

Definition at line 40 of file mo_mpr_global_variables.f90.

18.16.2.10 ntillhorizons

```
integer(i4), dimension(:), allocatable mo_mpr_global_variables::soiltype::ntillhorizons [private]
```

Definition at line 52 of file mo_mpr_global_variables.f90.

18.16.2.11 rzdepth

```
real(dp), dimension(:), allocatable mo_mpr_global_variables::soiltype::rzdepth [private]
```

Definition at line 49 of file mo_mpr_global_variables.f90.

18.16.2.12 sand

```
real(dp), dimension(:, :), allocatable mo_mpr_global_variables::soiltype::sand  [private]
```

Definition at line 46 of file mo_mpr_global_variables.f90.

18.16.2.13 thetafc

```
real(dp), dimension(:, :), allocatable mo_mpr_global_variables::soiltype::thetafc  [private]
```

Definition at line 63 of file mo_mpr_global_variables.f90.

18.16.2.14 thetafc_till

```
real(dp), dimension(:, :, :), allocatable mo_mpr_global_variables::soiltype::thetafc_till
[private]
```

Definition at line 61 of file mo_mpr_global_variables.f90.

18.16.2.15 thetapw

```
real(dp), dimension(:, :), allocatable mo_mpr_global_variables::soiltype::thetapw  [private]
```

Definition at line 66 of file mo_mpr_global_variables.f90.

18.16.2.16 thetapw_till

```
real(dp), dimension(:, :, :), allocatable mo_mpr_global_variables::soiltype::thetapw_till
[private]
```

Definition at line 64 of file mo_mpr_global_variables.f90.

18.16.2.17 thetas

```
real(dp), dimension(:, :), allocatable mo_mpr_global_variables::soiltype::thetas  [private]
```

Definition at line 57 of file mo_mpr_global_variables.f90.

18.16.2.18 thetas_till

```
real(dp), dimension(:, :, :), allocatable mo_mpr_global_variables::soiltype::thetas_till  [private]
```

Definition at line 55 of file mo_mpr_global_variables.f90.

18.16.2.19 ud

```
real(dp), dimension(:, :), allocatable mo_mpr_global_variables::soiltype::ud  [private]
```

Definition at line 43 of file mo_mpr_global_variables.f90.

18.16.2.20 wd

```
real(dp), dimension(:, :, :), allocatable mo_mpr_global_variables::soiltype::wd  [private]
```

Definition at line 50 of file mo_mpr_global_variables.f90.

The documentation for this type was generated from the following file:

- src/MPR/mo_mpr_global_variables.f90

18.17 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)

18.17.1 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

Definition at line 49 of file mo_spatial_agg_disagg_forcing.f90.

18.17.2 Member Function/Subroutine Documentation

18.17.2.1 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 )
```

Definition at line 96 of file mo_spatial_agg_disagg_forcing.f90.

References mo_common_constants::nodata_dp.

18.17.2.2 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 )
```

Definition at line 204 of file mo_spatial_agg_disagg_forcing.f90.

References mo_common_constants::nodata_dp.

The documentation for this interface was generated from the following file:

- src/mHM/mo_spatial_agg_disagg_forcing.f90

18.18 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)

18.18.1 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

Definition at line 84 of file mo_spatial_agg_disagg_forcing.f90.

18.18.2 Member Function/Subroutine Documentation

18.18.2.1 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 )
```

Definition at line 315 of file mo_spatial_agg_disagg_forcing.f90.

References mo_common_constants::nodata_dp.

18.18.2.2 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 )
```

Definition at line 381 of file mo_spatial_agg_disagg_forcing.f90.

References mo_common_constants::nodata_dp.

The documentation for this interface was generated from the following file:

- src/mHM/mo_spatial_agg_disagg_forcing.f90

18.19 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)

18.19.1 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

Definition at line 59 of file mo_mpr_restart.f90.

18.19.2 Member Function/Subroutine Documentation

18.19.2.1 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]
```

Definition at line 413 of file mo_mpr_restart.f90.

18.19.2.2 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]
```

Definition at line 368 of file mo_mpr_restart.f90.

18.19.2.3 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]
```

Definition at line 458 of file mo_mpr_restart.f90.

18.19.2.4 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]
```

Definition at line 515 of file mo_mpr_restart.f90.

The documentation for this interface was generated from the following file:

- src/MPR/mo_mpr_restart.f90

18.20 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)

18.20.1 Detailed Description

TODO: add description.

TODO: add description

Parameters

in,out	*type(NcDataset) :: nc*	NcDataset to add variable to

Parameters

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

Definition at line 60 of file mo_restart.f90.

18.20.2 Member Function/Subroutine Documentation

18.20.2.1 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]
```

Definition at line 830 of file mo_restart.f90.

18.20.2.2 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]
```

Definition at line 785 of file mo_restart.f90.

18.20.2.3 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]
```

Definition at line 875 of file mo_restart.f90.

18.20.2.4 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]
```

Definition at line 932 of file mo_restart.f90.

The documentation for this interface was generated from the following file:

- src/mHM/mo_restart.f90

Chapter 19

File Documentation

19.1 doc/1-main.dox File Reference

19.2 doc/2-get_started.dox File Reference

19.3 doc/3-data_preparation.dox File Reference

19.4 doc/4-visualise_out.dox File Reference

19.5 doc/5-calibration.dox File Reference

19.6 doc/6-style_guide.dox File Reference

19.7 doc/7-test_basin.dox File Reference

19.8 doc/8-protocols_for_setup_new_mHM_basin.dox File Reference

19.9 doc/9-mRM.dox File Reference

19.10 doc/DEPENDENCIES.md File Reference

19.11 doc/INSTALL.md File Reference

19.12 doc/mhm_papers.md File Reference

19.13 doc/RELEASES.md File Reference

19.14 src/common/mo_check.f90 File Reference

Input checking routines.

Modules

- module mo_check

Functions/Subroutines

- subroutine, public mo_check::check_dir (path, text_, throwError_, tab_, text_length_)

  Check if a given directory exists.

19.14.1 Detailed Description

Input checking routines.

This module provides sanity checks for the input data.

Authors

Sebastian Mueller

Date

Nov 2020

19.15 src/common/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::maxnodomains = 50_i4
- integer(i4), parameter, public mo_common_constants::maxnlcovers = 50_i4
- character(64), parameter, public mo_common_constants::soilhorizonsvarname = "L1_SoilHorizons"
- character(64), parameter, public mo_common_constants::landcoverperiodsvarname = "L1_LandCover↩ Periods"
- character(64), parameter, public mo_common_constants::laivarname = "L1_LAITimesteps"

19.16 src/common/mo_common_datetime_type.f90 File Reference

Data Types

- type mo_common_datetime_type::datetimeinfo

Modules

- module mo_common_datetime_type

Functions/Subroutines

- subroutine mo_common_datetime_type::datetimeinfo_init (this, iDomain)
- subroutine mo_common_datetime_type::datetimeinfo_increment (this)
- subroutine mo_common_datetime_type::datetimeinfo_update_lai_timestep (this)
- logical function mo_common_datetime_type::datetimeinfo_writeout (this, timeStep_model_outputs, tt)

19.17 src/common/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.

19.18 src/common/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.

19.19 src/common/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)

  Read main configurations commonly used by mHM, mRM and MPR.
- subroutine mo_common_read_config::init_domain_variable (nDomains, optiData, domainMeta)
- subroutine mo_common_read_config::init_domain_variable_for_master (domainMeta, colMasters, colDomain)
- subroutine mo_common_read_config::distributedomainsroundrobin (nproc, rank, domainMeta)
- subroutine mo_common_read_config::distribute_processes_to_domains_according_to_role (optiData, rank, domainMeta, colMasters, colDomain)

19.20 src/common/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.

19.21 src/common/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 domain
- subroutine, public mo_common_restart::read_grid_info (domainID, InFile, level_name, new_grid)

  reads configuration apart from Level 11 configuration from a restart directory

19.22 src/common/mo_common_variables.f90 File Reference

Data Types

- type mo_common_variables::period
- type mo_common_variables::grid
- type mo_common_variables::gridremapper
- type mo_common_variables::domain_meta

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_domain
- logical, public mo_common_variables::write_restart
- character(256), dimension(:), allocatable, public mo_common_variables::mhmfilerestartout
- character(256), dimension(:), allocatable, public mo_common_variables::mrmfilerestartout
- 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
- real(dp), dimension(:), allocatable, public mo_common_variables::l0_elev
- integer(i4), dimension(:, :), allocatable, public mo_common_variables::l0_lcover
- type(mpi_comm) mo_common_variables::comm
- type(domain_meta), public mo_common_variables::domainmeta
- integer(i4), public mo_common_variables::nuniquel0domains
- 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 = 11
- 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

19.23 src/common/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_domain_indices (grids, indices)

  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.

19.24 src/common/mo_read_latlon.f90 File Reference

Modules

- module mo_read_latlon

  reading latitude and longitude coordinates for each domain

Functions/Subroutines

- subroutine, public mo_read_latlon::read_latlon (ii, lon_var_name, lat_var_name, level_name, level)

  reads latitude and longitude coordinates

19.25 src/common/mo_read_nc.f90 File Reference

Modules

- module mo_read_nc

  Reads forcing input data.

Functions/Subroutines

- subroutine, public mo_read_nc::read_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_nc::read_const_nc (folder, nRows, nCols, varName, data, fileName)

  Reads time independent forcing input in NetCDF file format.
- subroutine, public mo_read_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_nc::get_time_vector_and_select (var, fname, inctimestep, time_start, time_cnt, target↩ _period)

  TODO: add description.

19.26 src/common/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.

19.27 src/common/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.

19.28 src/common/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

19.29 src/common_mHM_mRM/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)

19.30 src/common_mHM_mRM/mo_common_mHM_mRM_MPI_tools.f90 File Reference

Modules

- module mo_common_mhm_mrm_mpi_tools

  tools for MPI communication that are mHM or mRM specific

Functions/Subroutines

- subroutine, public mo_common_mhm_mrm_mpi_tools::distribute_parameterset (parameterset)
- subroutine, public mo_common_mhm_mrm_mpi_tools::get_parameterset (parameterset)

19.31 src/common_mHM_mRM/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.
- subroutine mo_common_mhm_mrm_read_config::period_copy_period_data (toPeriod, fromPeriod)

19.32 src/common_mHM_mRM/mo_common_mHM_mRM_restart.f90 File Reference

Data Types

- interface mo_common_mhm_mrm_restart::check_consistency_element

Modules

- module mo_common_mhm_mrm_restart

  TODO: add description.

Functions/Subroutines

- subroutine, public mo_common_mhm_mrm_restart::check_dimension_consistency (iBasin, nSoilHorizons↩ _temp, soilHorizonBoundaries_temp, nLAIs_temp, LAIBoundaries_temp, nLandCoverPeriods_temp, land↩ CoverPeriodBoundaries_temp)

  checks dimension configurations read from restart file
- subroutine mo_common_mhm_mrm_restart::check_consistency_element_dp (item1, item2, name, iBasin)
- subroutine mo_common_mhm_mrm_restart::check_consistency_element_i4 (item1, item2, name, iBasin)

19.33 src/common_mHM_mRM/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
- logical, public mo_common_mhm_mrm_variables::mrm_read_river_network
- 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::mhmfilerestartin
- character(256), dimension(:), allocatable, public mo_common_mhm_mrm_variables::mrmfilerestartin
- 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

19.34 src/common_mHM_mRM/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.

19.35 src/mHM/mhm_driver.f90 File Reference

Functions/Subroutines

- program mhm_driver

  Distributed precipitation-runoff model mHM.

19.35.1 Function/Subroutine Documentation

19.35.1.1 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 domains and a given period.

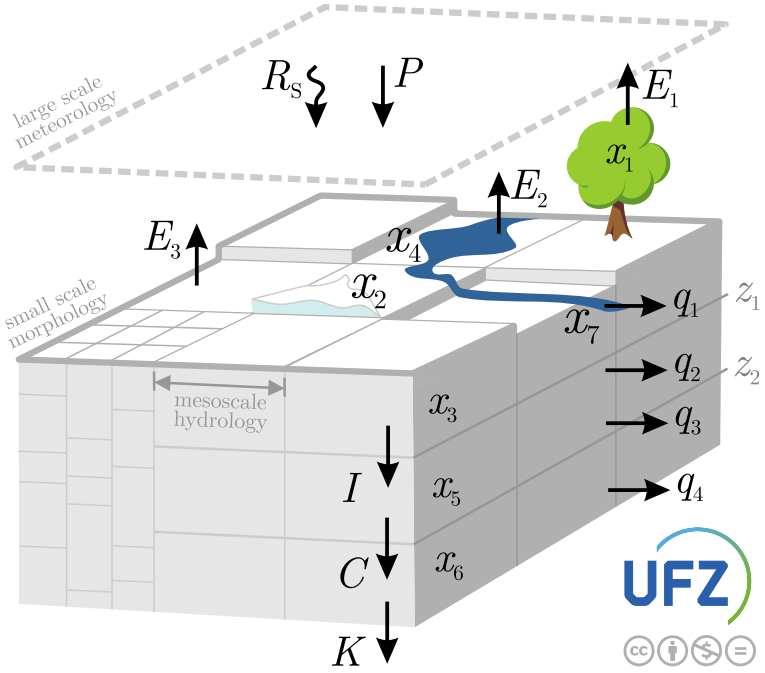


Figure 19.1: Typical mHM cell

Authors

Luis Samaniego & Rohini Kumar (UFZ)

Date

Jun 2018

Version

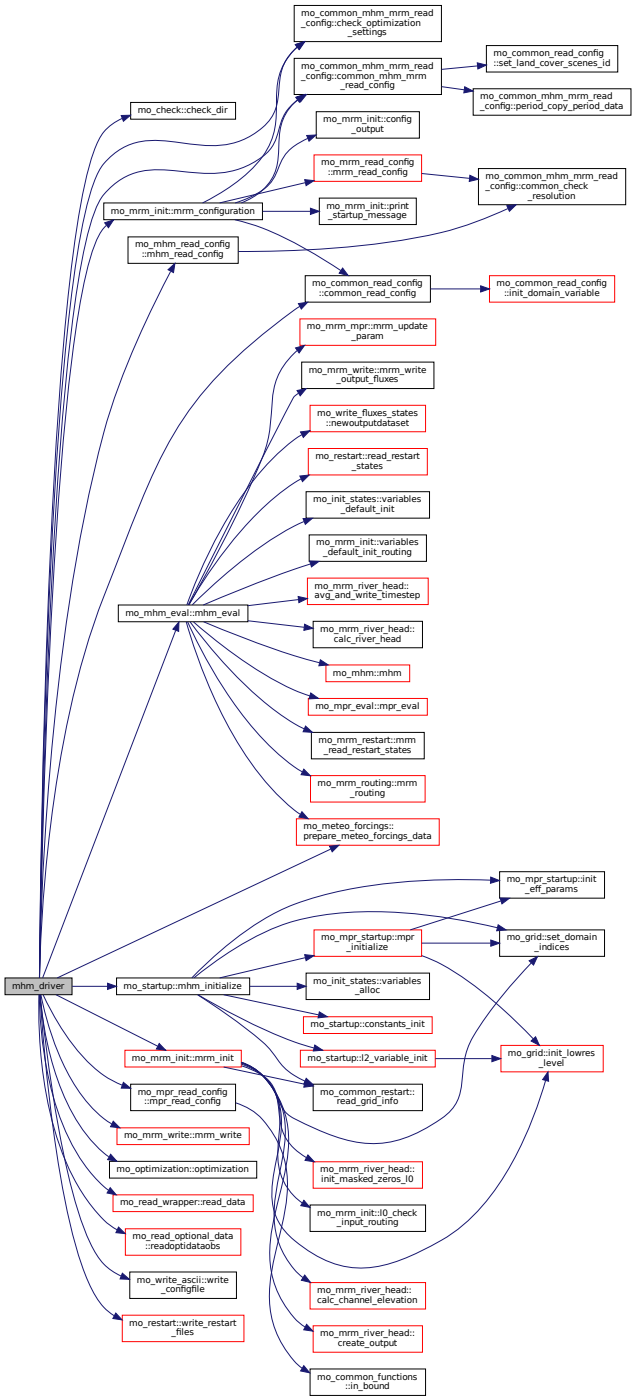
5.11.2

Copyright

(c) $2005 - 2021$, Helmholtz-Zentrum fuer Umweltforschung GmbH - UFZ. All rights reserved.

software should be clearly marked, so as not to confuse it with the version available from UFZ. This code can be used for research purposes ONLY provided that the following sources are acknowledged:
Samaniego L., Kumar R., Attinger S. (2010): Multiscale parameter regionalization of a grid-based hydrologic model at the mesoscale. Water Resour. Res., 46, W05523, doi:10.1029/2008WR007327.
Kumar, R., L. Samaniego, and S. Attinger (2013), Implications of distributed hydrologic model parameterization on water fluxes at multiple scales and locations, Water Resour. Res., 49, doi:10.1029/2012WR012195.
For commercial applications you have to consult the authorities of the UFZ.
Definition at line 79 of file mhm_driver.f90.
References mo_check::check_dir(), 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_file::file_main, 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_configuration(), mo_mrm_init::mrm_init(), mo_mrm_write::mrm_write(), mo_common_mhm↩_mrm_variables::ntstepday, mo_optimization::optimization(), mo_meteo_forcings::prepare_meteo_forcings_data(), mo_read_wrapper::read_data(), mo_read_optional_data::readoptidataobs(), 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:



19.36 src/mHM/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.

19.37 src/mHM/mo_file.f90 File Reference

Modules

- module mo_file

  Provides file names and units for mHM.

Variables

- character(len=∗), parameter mo_file::version = '5.11.2'

  Current mHM model version (will be set to 5.11.2)
- character(len=∗), parameter mo_file::version_date = 'Jul 2021'

  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.

19.38 src/mHM/mo_global_variables.f90 File Reference

Modules

- module mo_global_variables

  Global variables ONLY used in reading, writing and startup.

Variables

- integer(i4) mo_global_variables::output_deflate_level
- logical mo_global_variables::output_double_precision
- 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
- 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::dirradiation
- integer(i4), parameter, public mo_global_variables::routingstates = 2
- 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_ssrd
- real(dp), dimension(:, :), allocatable, public mo_global_variables::l1_strd
- real(dp), dimension(:, :), allocatable, public mo_global_variables::l1_tann
- real(dp), dimension(:, :), allocatable, public mo_global_variables::l1_sm
- logical, dimension(:, :), allocatable, public mo_global_variables::l1_sm_mask
- 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::nsoilhorizons_sm_input
- type(optidata), dimension(:), allocatable, public mo_global_variables::l1_smobs
- type(optidata), dimension(:), allocatable, public mo_global_variables::l1_neutronsobs
- type(optidata), dimension(:), allocatable, public mo_global_variables::l1_etobs
- type(optidata), dimension(:), allocatable, public mo_global_variables::l1_twsaobs
- 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(yearmonths_i4), public mo_global_variables::evap_coeff
- real(dp), dimension(yearmonths_i4), public mo_global_variables::fday_prec

- real(dp), dimension(yearmonths_i4), public mo_global_variables::fnight_prec
- real(dp), dimension(yearmonths_i4), public mo_global_variables::fday_pet
- real(dp), dimension(yearmonths_i4), public mo_global_variables::fnight_pet
- real(dp), dimension(yearmonths_i4), public mo_global_variables::fday_temp
- real(dp), dimension(yearmonths_i4), public mo_global_variables::fnight_temp
- real(dp), dimension(yearmonths_i4), public mo_global_variables::fday_ssrd
- real(dp), dimension(yearmonths_i4), public mo_global_variables::fnight_ssrd
- real(dp), dimension(yearmonths_i4), public mo_global_variables::fday_strd
- real(dp), dimension(yearmonths_i4), public mo_global_variables::fnight_strd
- real(dp), dimension(:), allocatable, public mo_global_variables::neutron_integral_afast

19.39 src/mHM/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.

19.40 src/mHM/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 (iDomain, domainID, tt)

  Prepare meteorological forcings data for a given variable.
- subroutine mo_meteo_forcings::meteo_forcings_wrapper (iDomain, dataPath, inputFormat, dataOut1, lower, upper, ncvarName)

  Prepare meteorological forcings data for mHM at Level-1.
- subroutine mo_meteo_forcings::meteo_weights_wrapper (iDomain, read_meteo_weights, dataPath, data↩Out1, lower, upper, ncvarName)

  Prepare weights for meteorological forcings data for mHM at Level-1.
- subroutine mo_meteo_forcings::chunk_config (iDomain, tt, read_flag, readPer)

  determines the start date, end date, and read_flag given Domain id and current timestep
- logical function mo_meteo_forcings::is_read (iDomain, tt)

  evaluate whether new chunk should be read at this timestep
- subroutine mo_meteo_forcings::chunk_size (iDomain, tt, readPer)

  calculate beginning and end of read Period, i.e. that is length of current chunk to read

19.41 src/mHM/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.

19.42 src/mHM/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::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

19.43 src/mHM/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, opti_domain_indices, runoff, smOptiSim, neutronsOptiSim, etOptiSim, twsOptiSim)

  Runs mhm with a specific parameter set and returns required variables, e.g. runoff.

19.44 src/mHM/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.

19.45 src/mHM/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.

19.46 src/mHM/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, public mo_objective_function::objective_master (parameterset, eval, arg1, arg2, arg3)

  Wrapper for objective functions.
- subroutine, public mo_objective_function::objective_subprocess (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, dimension(6) mo_objective_function::objective_q_et_tws_kge_catchment_avg (parameterset, eval)

  Objective function for et, tws and q.
- subroutine mo_objective_function::init_indexarray_for_opti_data (domainMeta, optidataOption, nOpti↩Domains, opti_domain_indices)

  creates an index array of the inidices of the domains eval should MPI process.
- 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 domain_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 domain_avg ET (standarized scores)
- subroutine mo_objective_function::create_domain_avg_tws (iDomain, twsOptiSim, tws_catch_avg_domain, tws_opti_catch_avg_domain, mask_times)
- subroutine mo_objective_function::create_domain_avg_et (iDomain, etOptiSim, et_catch_avg_domain, et_↩opti_catch_avg_domain, mask_times)
- subroutine mo_objective_function::convert_tws_to_twsa (twsOptiSim, L1_twsaObs, twsaOptiSim)

19.47 src/mHM/mo_pet.f90 File Reference

Modules

- module mo_pet

  Module for calculating reference/potential evapotranspiration [mm d-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, public mo_pet::extraterr_rad_approx (doy, latitude)

  Approximation of extraterrestrial radiation.
- elemental pure real(dp) function, public mo_pet::slope_satpressure (tavg)

  slope of saturation vapour pressure curve
- elemental pure real(dp) function, public mo_pet::sat_vap_pressure (tavg)

  calculation of the saturation vapour pressure

19.48 src/mHM/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::readoptidataobs (iDomain, domainID, L1_optiObs)

  Read evapotranspiration data from NetCDF file for calibration.

19.49 src/mHM/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 (OutFile)

  write restart files for each domain
- subroutine, public mo_restart::read_restart_states (iDomain, domainID, InFile)

  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)

19.50 src/mHM/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

19.51 src/mHM/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.

19.52 src/mHM/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, public 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, public 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

19.53 src/mHM/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)

19.54 src/mHM/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 (iDomain, level0_iDomain, level2_iDomain)

  Initalize Level-2 meteorological forcings data.

19.55 src/mHM/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.

- elemental subroutine, public mo_temporal_disagg_forcing::temporal_disagg_meteo_weights (meteo_val_↩ day, meteo_val_weights, meteo_val, weights_correction)

  Temporally distribute daily mean forcings onto time step.
- elemental subroutine, public mo_temporal_disagg_forcing::temporal_disagg_flux_daynight (isday, ntimesteps↩ _day, meteo_val_day, fday_meteo_val, fnight_meteo_val, meteo_val)

  Temporally distribute daily mean forcings onto time step.
- elemental subroutine, public mo_temporal_disagg_forcing::temporal_disagg_state_daynight (isday, ntimesteps_day, meteo_val_day, fday_meteo_val, fnight_meteo_val, meteo_val, add_correction)

  Temporally distribute daily mean state forcings onto time step.

19.56 src/mHM/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.

19.57 src/mHM/mo_write_fluxes_states.f90 File Reference

Data Types

- interface mo_write_fluxes_states::outputvariable
- 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 (iDomain, 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 (iDomain)

  Create and initialize output file for X & Y coordinate system.
- subroutine mo_write_fluxes_states::writevariableattributes (var, long_name, unit)

  Write output variable attributes.
- character(16) function mo_write_fluxes_states::fluxesunit (iDomain)

  Generate a unit string.

19.58 src/MPR/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

19.59 src/MPR/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, opti_domain_indices)

  Runs MPR and writes to global effective parameters.

19.60 src/MPR/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.

19.61 src/MPR/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
- real(dp), dimension(:), allocatable, public mo_mpr_global_variables::laiboundaries
- 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

19.62 src/MPR/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.

19.63 src/MPR/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.

19.64 src/MPR/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 (OutFile)

  write restart files for each domain
- 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)

19.65 src/MPR/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

19.66 src/MPR/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

19.67 src/MPR/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, processMatrix, 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

19.68 src/MPR/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 (iDomain)

  Check for errors in L0 input data.
- subroutine mo_mpr_startup::l0_variable_init (iDomain)

  level 0 variable initialization
- subroutine, public mo_mpr_startup::init_eff_params (ncells1)

  Allocation of space for mHM related L1 and L11 variables.

19.69 src/MPR/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.

19.70 src/MPR/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 (iDomain, nrows, ncols, mask, LAIPer_iDomain)

  Prepare gridded daily LAI data.
- subroutine, public mo_prepare_gridded_lai::prepare_gridded_mean_monthly_lai_data (iDomain, nrows, ncols, mask)

  prepare_gridded_mean_monthly_LAI_data

19.71 src/MPR/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.

19.72 src/MPR/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.

19.73 src/MPR/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.

19.74 src/MPR/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_fine↩ Scale_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, n↩ TCells0_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

19.75 src/MPR/mpr_driver.f90 File Reference

Modules

- module dummy_mpr

Functions/Subroutines

- program mpr_driver

  Distributed precipitation-runoff model mHM.

19.75.1 Function/Subroutine Documentation

19.75.1.1 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 domains and a given period.

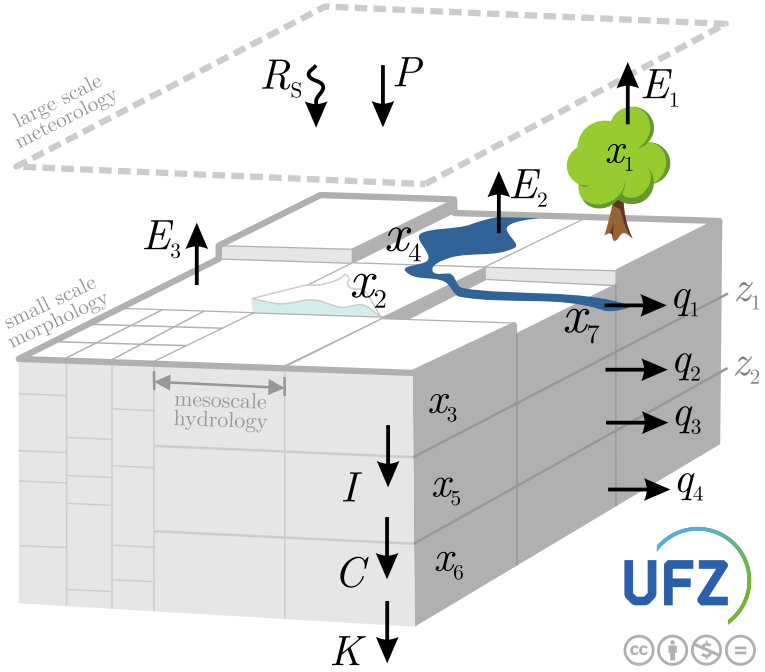


Figure 19.2: Typical mHM cell

Authors

Luis Samaniego & Rohini Kumar (UFZ)

Date

Dec 2015

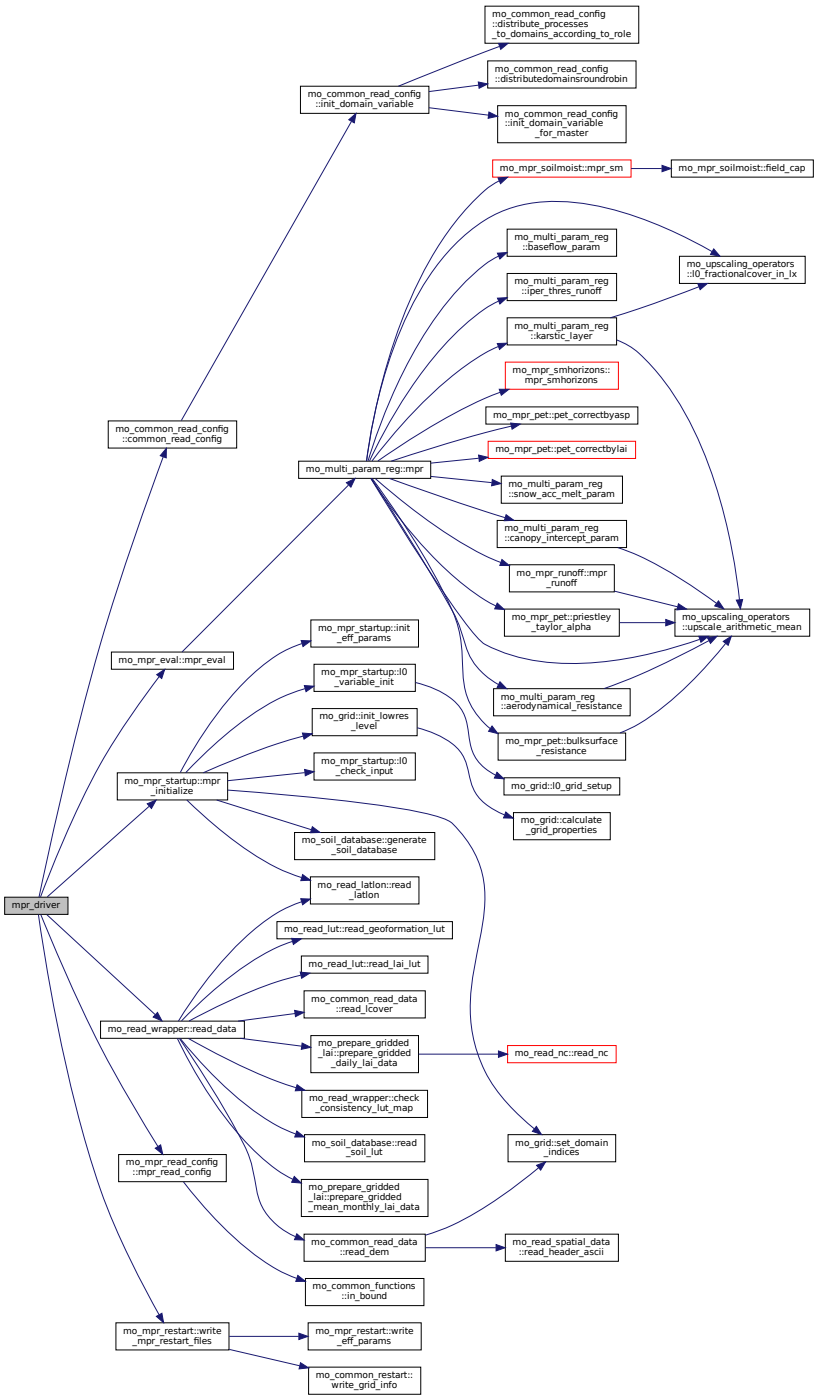
Version

0.1

Copyright

Definition at line 59 of file mpr_driver.f90.

References mo_common_read_config::common_read_config(), mo_mpr_file::file_namelist_mpr_param, mo_↩ common_variables::mhmfilerestartout, mo_mpr_eval::mpr_eval(), mo_mpr_startup::mpr_initialize(), mo_mpr_↩ read_config::mpr_read_config(), mo_read_wrapper::read_data(), 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:



19.76 src/mRM/mo_mrm_constants.f90 File Reference

Modules

- module mo_mrm_constants

  Provides mRM specific constants.

Variables

- integer(i4), parameter, public mo_mrm_constants::noutflxstate = 2_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/)

19.77 src/mRM/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

19.78 src/mRM/mo_mrm_global_variables.f90 File Reference

Data Types

- type mo_mrm_global_variables::gaugingstation
- type mo_mrm_global_variables::domaininfo_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::output_deflate_level_mrm
- logical mo_mrm_global_variables::output_double_precision_mrm
- 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::ngaugeslocal
- 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(domaininfo_mrm), dimension(:), allocatable, target, public mo_mrm_global_variables::domain_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
- type(riv_temp_type), public mo_mrm_global_variables::riv_temp_pcs

  This is a container for the river temperature routing process (pcs)

19.79 src/mRM/mo_mrm_init.f90 File Reference

Modules

- module mo_mrm_init

  Wrapper for initializing Routing.

Functions/Subroutines

- subroutine, public mo_mrm_init::mrm_configuration (file_namelist, unamelist, file_namelist_param, unamelist_param, ReadLatLon)
- subroutine, public mo_mrm_init::mrm_init (file_namelist, unamelist, file_namelist_param, unamelist_param, ReadLatLon)

  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 (L0Domain_iDomain)

  TODO: add description.
- subroutine mo_mrm_init::variables_alloc_routing (iDomain)

  TODO: add description.

19.80 src/mRM/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 (iDomain, param)

  TODO: add description.
- subroutine, public mo_mrm_mpr::mrm_update_param (iDomain, param)

  TODO: add description.

19.81 src/mRM/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 (iDomain)

  TODO: add description.
- subroutine, public mo_mrm_net_startup::l11_flow_direction (iDomain)

  Determine the flow direction of the upscaled river network at level L11.
- subroutine, public mo_mrm_net_startup::l11_set_network_topology (iDomain)

  Set network topology.
- subroutine, public mo_mrm_net_startup::l11_routing_order (iDomain)

  Find routing order, headwater cells and sink.
- subroutine, public mo_mrm_net_startup::l11_link_location (iDomain)

  Estimate the LO (row,col) location for each routing link at level L11.
- subroutine, public mo_mrm_net_startup::l11_set_drain_outlet_gauges (iDomain)

  Draining cell identification and Set gauging node.
- subroutine, public mo_mrm_net_startup::l11_stream_features (iDomain)

  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::movedownnonecell (fDir, iRow, jCol)

  TODO: add description.
- subroutine mo_mrm_net_startup::celllength (iDomain, 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 (iDomain)

  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 (iDomain, param)

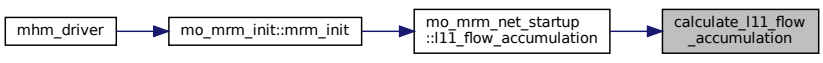
  L11 celerity based on L0 elevation and L0 fAcc.

19.81.1 Function/Subroutine Documentation

19.81.1.1 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 )
```

Definition at line 2075 of file mo_mrm_net_startup.f90.
Referenced by mo_mrm_net_startup::l11_flow_accumulation().
Here is the caller graph for this function:



19.82 src/mRM/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.
- real(dp) function, public mo_mrm_objective_function_runoff::single_objective_runoff_master (parameterset, eval, arg1, arg2, arg3)

  Wrapper for objective functions optimizing agains runoff.
- subroutine, public mo_mrm_objective_function_runoff::single_objective_runoff_subprocess (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.
- real(dp) function mo_mrm_objective_function_runoff::objective_sse_boxcox (parameterset, eval)

  Objective function of sum of squared errors of transformed streamflow.
- subroutine, public mo_mrm_objective_function_runoff::extract_runoff (gaugeId, runoff, runoff_agg, runoff_↩obs, runoff_obs_mask)

  extracts runoff data from global variables

19.83 src/mRM/mo_mrm_pre_routing.f90 File Reference

Modules

- module mo_mrm_pre_routing

  Performs pre-processing for routing for mHM at level L11.

Functions/Subroutines

- subroutine, public mo_mrm_pre_routing::l11_runoff_acc (qAll, efecArea, L1_L11_Id, L11_areaCell, L11_↩L1_Id, TS, map_flag, qAcc)

  total runoff accumulation at L11.
- subroutine, public mo_mrm_pre_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_pre_routing::l11_e_acc (qAll, efecArea, L1_L11_Id, L11_areaCell, L11_L1_Id, TS, map_flag, qAcc)

  temperature energy accumulation at L11.
- subroutine, public mo_mrm_pre_routing::calc_l1_runoff_e (fSealed_area_fraction, fast_interflow, slow_↩interflow, baseflow, direct_runoff, temp_air, mean_temp_air, lateral_E)

  calculate lateral temperature energy from runoff components.
- subroutine, public mo_mrm_pre_routing::l11_meteo_acc (meteo_all, efecArea, L1_L11_Id, L11_areaCell, L11_L1_Id, map_flag, meteo_acc)

  meteo forcing accumulation at L11 for temperature routing.

19.84 src/mRM/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.

19.85 src/mRM/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 (iDomain)

  read simulated runoff that is to be routed
- subroutine, public mo_mrm_read_data::mrm_read_bankfull_runoff (iDomain)
- subroutine mo_mrm_read_data::rotate_fdir_variable (x)

  TODO: add description.

19.86 src/mRM/mo_mrm_restart.f90 File Reference

Modules

- module mo_mrm_restart

  Restart routines.

Functions/Subroutines

- subroutine, public mo_mrm_restart::mrm_write_restart (iDomain, domainID, OutFile)

  write routing states and configuration
- subroutine, public mo_mrm_restart::mrm_read_restart_states (iDomain, domainID, InFile)

  read routing states
- subroutine, public mo_mrm_restart::mrm_read_restart_config (iDomain, domainID, InFile)

  reads Level 11 configuration from a restart directory

19.87 src/mRM/mo_mrm_riv_temp_class.f90 File Reference

Data Types

- module mo_mrm_riv_temp_class::riv_temp_type

  This is a container to define the river temperature routing in the current time step.

Modules

- module mo_mrm_riv_temp_class

  Class for the river temperature calculations.

Functions/Subroutines

- subroutine mo_mrm_riv_temp_class::config (self, file_namelist, unamelist, file_namelist_param, unamelist↩ _param)

  configure the riv_temp_type class from the mhm namelist
- subroutine mo_mrm_riv_temp_class::init (self, nCells)

  initalize the riv_temp_type class for the current domain

- subroutine mo_mrm_riv_temp_class::init_area (self, iDomain, L11_netPerm, L11_fromN, L11_length, nLinks, nCells, nrows, ncols, L11_mask)

  initialize the river area of riv_temp_type class for the current domain
- subroutine mo_mrm_riv_temp_class::init_riv_temp (self, time, ntimesteps_day, temp_air, read_meteo_↩ weights, temp_weights, fday_temp, fnight_temp, efecarea, L1_L11_Id, L11_areacell, L11_L1_Id, map_flag)

  initialize the river temperature of riv_temp_type class for the current domain
- subroutine mo_mrm_riv_temp_class::reset_timestep (self)

  reset riv_temp_type class for next timestep
- subroutine mo_mrm_riv_temp_class::alloc_lateral (self, nCells)

  allocate lateral temp components of riv_temp_type class for current domain
- subroutine mo_mrm_riv_temp_class::dealloc_lateral (self)

  deallocate lateral temp components of riv_temp_type
- subroutine mo_mrm_riv_temp_class::acc_source_e (self, time, ntimesteps_day, fSealed_area_fraction, fast_interflow, slow_interflow, baseflow, direct_runoff, temp_air, mean_temp_air, ssrd_day, strd_day, read_↩ meteo_weights, temp_weights, fday_temp, fnight_temp, fday_ssrd, fnight_ssrd, fday_strd, fnight_strd)

  accumulate energy sources of riv_temp_type
- subroutine mo_mrm_riv_temp_class::finalize_source_e (self, efecarea, L1_L11_Id, L11_areacell, L11_L1_↩ Id, timestep, map_flag)

  finalize energy sources of riv_temp_type
- real(dp) function mo_mrm_riv_temp_class::get_lrad_out (self, riv_temp)

  get outgoing longwave radiation of riv_temp_type
- real(dp) function mo_mrm_riv_temp_class::get_lat_heat (self, air_temp, netrad)

  latent heat flux of riv_temp_type
- real(dp) function mo_mrm_riv_temp_class::get_sens_heat (self, air_temp, riv_temp)

  sensible heat flux of riv_temp_type
- real(dp) function mo_mrm_riv_temp_class::get_e_io (self, riv_temp, cell)

  get complete energy source of riv_temp_type at given cell
- subroutine mo_mrm_riv_temp_class::l11_routing_e (self, nLinks, netPerm, netLink_fromN, netLink_toN, netLink_C1, netLink_C2, nInflowGauges, InflowHeadwater, InflowNodeList, L11_qTR, L11_Qmod)

  execute the temperature routing of riv_temp_type
- subroutine mo_mrm_riv_temp_class::init_iter (self)

  initialize iterative solver of riv_temp_type
- subroutine mo_mrm_riv_temp_class::next_iter (self, T_est, T_rout)

  execute next iteration with iterative solver of riv_temp_type

19.88 src/mRM/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 (iDomain, data)
- subroutine, private mo_mrm_river_head::reset_sum (iDomain, data)
- subroutine, public mo_mrm_river_head::calc_channel_elevation ()
- subroutine, public mo_mrm_river_head::calc_river_head (iDomain, L11_Qmod, river_head)
- real(dp) function mo_mrm_river_head::calc_slope (iDomain, elev0, fDir0, i, j)
- subroutine, public mo_mrm_river_head::avg_and_write_timestep (iDomain, timestep, data)
- subroutine, public mo_mrm_river_head::create_output (iDomain, 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

19.89 src/mRM/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, InflowGauge↩ Headwater, 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_routing (nNodes, nLinks, netPerm, netLink_fromN, netLink_toN, netLink↩ _C1, netLink_C2, netNode_qOUT, nInflowGauges, InflowHeadwater, InflowNodeList, netNode_qTIN, net↩ Node_qTR, netNode_Qmod)

  Performs runoff routing for mHM at L11 upscaled network (Routing Network).

19.90 src/mRM/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, domain_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.

19.91 src/mRM/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 (iDomain, nCells, timeStep_model_outputs, domainDateTime, tt, 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

- type(outputdataset) mo_mrm_write::nc

19.92 src/mRM/mo_mrm_write_fluxes_states.f90 File Reference

Data Types

- interface mo_mrm_write_fluxes_states::outputvariable
- 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 (iDomain, mask, nCells)

Initialize OutputDataset.

- subroutine mo_mrm_write_fluxes_states::updatedataset (self, sidx, eidx, L11_Qmod, L11_riv_temp)

  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 (iDomain)

  Create and initialize output file for X & Y coordinate system.

- subroutine mo_mrm_write_fluxes_states::writevariableattributes (var, long_name, unit)

  Write output variable attributes.

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. 332, 339, 340, 531

[8] R Courant, K Friedrichs, and H Lewy. Über die partiellen Differenzengleichungen der mathematischen Physik. *Mathematische Annalen*, 100(1):32–74, 1928. 46

[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. 47

[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. 268

[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, 47

[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

acc_source_e
    mo_mrm_riv_temp_class, 324
    mo_mrm_riv_temp_class::riv_temp_type, 525
active
    mo_mrm_riv_temp_class::riv_temp_type, 532
add_inflow
    mo_mrm_pre_routing, 307
aerodynamical_resistance
    mo_multi_param_reg, 367
albedo_water
    mo_mrm_riv_temp_class::riv_temp_type, 532
all
    mo_mrm_write_fluxes_states::outputdataset, 503
    mo_write_fluxes_states::outputdataset, 507
alloc_lateral
    mo_mrm_riv_temp_class, 325
    mo_mrm_riv_temp_class::riv_temp_type, 526
alma_convention
    mo_common_variables, 117
approx_mon_int
    mo_neutrons, 378
approx_mon_int_eps
    mo_neutrons, 379
approx_mon_int_steps
    mo_neutrons, 380
autocorrelation
    mo_mrm_signatures, 344
average
    mo_mrm_write_fluxes_states::outputvariable, 510
    mo_write_fluxes_states::outputvariable, 515
avg
    mo_mrm_write_fluxes_states::outputvariable, 510
    mo_write_fluxes_states::outputvariable, 515
avg_and_write_timestep
    mo_mrm_river_head, 333

baseflow_param
    mo_multi_param_reg, 368
before
    mo_mrm_write_fluxes_states::outputvariable, 510
    mo_write_fluxes_states::outputvariable, 515
between
    mo_mrm_write_fluxes_states::outputvariable, 511
    mo_write_fluxes_states::outputvariable, 515
bisect_iter
    mo_mrm_riv_temp_class::riv_temp_type, 532
bulkdens_orgmatter
    mo_mpr_constants, 175
bulksurface_resistance
    mo_mpr_pet, 201

c1_initstatesm
    mo_mhm_constants, 164
    mo_mpr_constants, 175
c2tstu
    mo_common_mhm_mrm_variables, 101
calc_channel_elevation
    mo_mrm_river_head, 334
calc_l1_runoff_e
    mo_mrm_pre_routing, 308
calc_river_head
    mo_mrm_river_head, 335
calc_slope
    mo_mrm_river_head, 335
calculate_grid_properties
    mo_grid, 142
calculate_l11_flow_accumulation
    mo_mrm_net_startup.f90, 587
calls
    mo_mrm_write_fluxes_states::outputvariable, 511
    mo_write_fluxes_states::outputvariable, 515
canopy_interc
    mo_canopy_interc, 83
canopy_intercept_param
    mo_multi_param_reg, 369
case
    mo_mrm_riv_temp_class::riv_temp_type, 532
cellarea
    mo_common_variables::grid, 496
cellcoor
    mo_common_variables::grid, 497
celllength
    mo_mrm_net_startup, 262
cellsize
    mo_common_variables::grid, 497
check_consistency_element_dp
    mo_common_mhm_mrm_restart, 99
    mo_common_mhm_mrm_restart::check_consistency_element, 487
check_consistency_element_i4
    mo_common_mhm_mrm_restart, 100
    mo_common_mhm_mrm_restart::check_consistency_element, 487
check_consistency_lut_map
    mo_read_wrapper, 436
check_dimension_consistency
    mo_common_mhm_mrm_restart, 100
check_dir
    mo_check, 84
check_optimization_settings

mo_common_mhm_mrm_read_config, 96
chunk_config
mo_meteo_forcings, 151
chunk_size
mo_meteo_forcings, 152
circum
mo_template, 460
clay
mo_mpr_global_variables::soiltype, 538
close
mo_mrm_write_fluxes_states, 359
mo_mrm_write_fluxes_states::outputdataset, 502
mo_write_fluxes_states, 478
mo_write_fluxes_states::outputdataset, 506
comlocal
mo_common_variables::domain_meta, 491
comm
mo_common_variables, 117
commaster
mo_common_variables::domain_meta, 492
common_check_resolution
mo_common_mhm_mrm_read_config, 96
common_mhm_mrm_read_config
mo_common_mhm_mrm_read_config, 97
common_read_config
mo_common_read_config, 107
config
mo_mrm_riv_temp_class, 326
mo_mrm_riv_temp_class::riv_temp_type, 526
config_output
mo_mrm_init, 251
constants_init
mo_startup, 457
contact
mo_common_variables, 117
contains
mo_mrm_write_fluxes_states::outputvariable, 511
mo_write_fluxes_states::outputvariable, 516
conventions
mo_common_variables, 118
convert_tws_to_twsa
mo_objective_function, 387
cosmic
mo_neutrons, 381
cosmic_alpha
mo_mhm_constants, 164
cosmic_bd
mo_mhm_constants, 164
cosmic_l1
mo_mhm_constants, 164
cosmic_l2
mo_mhm_constants, 164
cosmic_l3
mo_mhm_constants, 164
cosmic_l4
mo_mhm_constants, 164
cosmic_n
mo_mhm_constants, 165
cosmic_vwclat
mo_mhm_constants, 165
count
mo_mrm_write_fluxes_states::outputdataset, 503
mo_mrm_write_fluxes_states::outputvariable, 511
mo_write_fluxes_states::outputdataset, 507
mo_write_fluxes_states::outputvariable, 516
counter
mo_mrm_write_fluxes_states::outputdataset, 503
mo_mrm_write_fluxes_states::outputvariable, 511
mo_write_fluxes_states::outputdataset, 507
mo_write_fluxes_states::outputvariable, 516
create_domain_avg_et
mo_objective_function, 388
create_domain_avg_tws
mo_objective_function, 388
create_output
mo_mrm_river_head, 336
created
mo_mrm_write_fluxes_states::outputdataset, 503
mo_write_fluxes_states::outputdataset, 507
createoutputfile
mo_mrm_write_fluxes_states, 359
mo_write_fluxes_states, 478

data
mo_mrm_write_fluxes_states::outputvariable, 511
mo_write_fluxes_states::outputvariable, 516
datetimeinfo_increment
mo_common_datetime_type, 89
datetimeinfo_init
mo_common_datetime_type, 89
datetimeinfo_update_lai_timestep
mo_common_datetime_type, 89
datetimeinfo_writeout
mo_common_datetime_type, 89
day
mo_common_datetime_type::datetimeinfo, 489
db
mo_mpr_global_variables::soiltype, 538
dbm
mo_mpr_global_variables::soiltype, 538
dds_r
mo_common_mhm_mrm_variables, 101
dealloc_lateral
mo_mrm_riv_temp_class, 326
mo_mrm_riv_temp_class::riv_temp_type, 527
delta_iter
mo_mrm_riv_temp_class::riv_temp_type, 532
delta_t
mo_mrm_riv_temp_class::riv_temp_type, 533
deltah
mo_mrm_constants, 228
dend
mo_common_variables::period, 519
depth
mo_mpr_global_variables::soiltype, 538
desilets_a0
mo_mhm_constants, 165

desilets_a1
mo_mhm_constants, 165
desilets_a2
mo_mhm_constants, 165
desiletsn0
mo_neutrons, 382
dir_riv_widths
mo_mrm_riv_temp_class::riv_temp_type, 533
dirabsvappressure
mo_global_variables, 128
dirbankfullrunoff
mo_mrm_global_variables, 237
dircommonfiles
mo_common_variables, 118
dirconfigout
mo_common_variables, 118
dirgauges
mo_mrm_global_variables, 237
dirgridded_lai
mo_mpr_global_variables, 191
dirlcover
mo_common_variables, 118
dirmaxtemperature
mo_global_variables, 129
dirmintemperature
mo_global_variables, 129
dirmorpho
mo_common_variables, 118
dirnetradiation
mo_global_variables, 129
dirout
mo_common_variables, 118
dirprecipitation
mo_global_variables, 129
dirradiation
mo_global_variables, 129
dirreferenceet
mo_global_variables, 130
dirtemperature
mo_global_variables, 130
dirtotalrunoff
mo_mrm_global_variables, 237
dirwindspeed
mo_global_variables, 130
distribute_parameterset
mo_common_mhm_mrm_mpi_tools, 94
distribute_processes_to_domains_according_to_role
mo_common_read_config, 109
distributedomainsroundrobin
mo_common_read_config, 109
doc/1-main.dox, 549
doc/2-get_started.dox, 549
doc/3-data_preparation.dox, 549
doc/4-visualise_out.dox, 549
doc/5-calibration.dox, 549
doc/6-style_guide.dox, 549
doc/7-test_basin.dox, 549
doc/8-protocols_for_setup_new_mHM_basin.dox, 549
doc/9-mRM.dox, 549
doc/DEPENDENCIES.md, 549
doc/INSTALL.md, 549
doc/mhm_papers.md, 549
doc/RELEASES.md, 549
domain
mo_mrm_write_fluxes_states::outputdataset, 503
mo_write_fluxes_states::outputdataset, 507
domain_mrm
mo_mrm_global_variables, 237
domainid
mo_mrm_global_variables::gaugingstation, 495
domainmeta
mo_common_variables, 119
dorouting
mo_common_variables::domain_meta, 492
dstart
mo_common_variables::period, 519
duffiedelta1
mo_mhm_constants, 165
duffiedelta2
mo_mhm_constants, 166
duffiedr
mo_mhm_constants, 166
dummy_mpr, 83

e11
mo_mrm_riv_temp_class::riv_temp_type, 533
emissivity_water
mo_mrm_riv_temp_class::riv_temp_type, 533
eps_dp
mo_common_constants, 86
eps_sp
mo_common_constants, 86
evalper
mo_common_mhm_mrm_variables, 102
evap_coeff
mo_global_variables, 130
extract_runoff
mo_mrm_objective_function_runoff, 278
extraterr_rad_approx
mo_pet, 411

fday_pet
mo_global_variables, 130
fday_prec
mo_global_variables, 130
fday_ssrd
mo_global_variables, 131
fday_strd
mo_global_variables, 131
fday_temp
mo_global_variables, 131
feddes_et_reduction
mo_soil_moisture, 451
field_cap
mo_mpr_soilmoist, 217
field_cap_c1
mo_mpr_constants, 175

field_cap_c2
mo_mpr_constants, 175
file_aspect
mo_mpr_file, 184
file_config
mo_common_file, 90
mo_mrm_file, 230
file_daily_discharge
mo_mrm_file, 231
file_defoutput
mo_file, 125
mo_mrm_file, 231
file_dem
mo_common_file, 90
file_facc
mo_mrm_file, 231
file_fdir
mo_mrm_file, 231
file_gaugeloc
mo_mrm_file, 231
file_geolut
mo_mpr_file, 184
file_gw_output
mo_mrm_file, 232
file_hydrogeoclass
mo_mpr_file, 185
file_laiclass
mo_mpr_file, 185
file_lailut
mo_mpr_file, 185
file_main
mo_file, 125
mo_mpr_file, 185
mo_mrm_file, 232
file_meteo_binary_end
mo_mpr_file, 185
file_meteo_header
mo_mpr_file, 186
file_mrm_output
mo_mrm_file, 232
file_namelist_mhm
mo_file, 125
file_namelist_mhm_param
mo_file, 126
file_namelist_mpr
mo_mpr_file, 186
file_namelist_mpr_param
mo_mpr_file, 186
file_namelist_mrm
mo_mrm_file, 232
file_namelist_param_mrm
mo_mrm_file, 232
file_opti
mo_common_mhm_mrm_file, 93
file_opti_nml
mo_common_mhm_mrm_file, 93
file_slope
mo_mpr_file, 186
mo_mrm_file, 232
file_soil_database
mo_mpr_file, 186
file_soil_database_1
mo_mpr_file, 186
file_soilclass
mo_mpr_file, 187
filelatlon
mo_common_variables, 119
filenametotalrunoff
mo_mrm_global_variables, 238
finalize_source_e
mo_mrm_riv_temp_class, 326
mo_mrm_riv_temp_class::riv_temp_type, 527
first_iter
mo_mrm_riv_temp_class::riv_temp_type, 533
flowdurationcurve
mo_mrm_signatures, 345
fluxesunit
mo_write_fluxes_states, 479
fname
mo_mrm_global_variables::gaugingstation, 495
fnight_pet
mo_global_variables, 131
fnight_prec
mo_global_variables, 131
fnight_ssrd
mo_global_variables, 131
fnight_strd
mo_global_variables, 131
fnight_temp
mo_global_variables, 132
fracsealed_cityarea
mo_mpr_global_variables, 191

gauge
mo_mrm_global_variables, 238
gaugeid
mo_mrm_global_variables::gaugingstation, 495
gaugeidlist
mo_mrm_global_variables::domaininfo_mrm, 493
gaugeindexlist
mo_mrm_global_variables::domaininfo_mrm, 493
gaugenodelist
mo_mrm_global_variables::domaininfo_mrm, 493
generate_soil_database
mo_soil_database, 448
genuchten
mo_mpr_soilmoist, 218
geocoordinates
mo_grid, 143
geounitkar
mo_mpr_global_variables, 191
geounitlist
mo_mpr_global_variables, 191
get_distance_two_lat_lon_points
mo_mrm_net_startup, 264
get_e_io
mo_mrm_riv_temp_class, 327

mo_mrm_riv_temp_class::riv_temp_type, 527
get_lat_heat
mo_mrm_riv_temp_class, 328
mo_mrm_riv_temp_class::riv_temp_type, 528
get_lrad_out
mo_mrm_riv_temp_class, 328
mo_mrm_riv_temp_class::riv_temp_type, 528
get_parameterset
mo_common_mhm_mrm_mpi_tools, 95
get_sens_heat
mo_mrm_riv_temp_class, 329
mo_mrm_riv_temp_class::riv_temp_type, 529
get_time_vector_and_select
mo_read_nc, 425
given_ts
mo_mrm_constants, 228
global_parameters
mo_common_variables, 119
global_parameters_name
mo_common_variables, 120
gw_coupling
mo_mrm_global_variables, 238

h2odens
mo_mhm_constants, 166
harsamconst
mo_mhm_constants, 166
high_res_grid
mo_common_variables::gridremapper, 499
history
mo_common_variables, 120
horizondepth_mhm
mo_mpr_global_variables, 191
hour
mo_common_datetime_type::datetimeinfo, 489
hydro_cond
mo_mpr_soilmoist, 219

id
mo_common_variables::grid, 497
mo_mpr_global_variables::soiltype, 539
mo_mrm_write_fluxes_states::outputdataset, 504
mo_write_fluxes_states::outputdataset, 507
idomain
mo_mrm_write_fluxes_states::outputdataset, 504
mo_write_fluxes_states::outputdataset, 507
iend
mo_common_variables::grid, 497
iflag_cordinate_sys
mo_common_variables, 120
iflag_soildb
mo_mpr_global_variables, 192
ilai
mo_common_datetime_type::datetimeinfo, 489
in_bound
mo_common_functions, 92
increment
mo_common_datetime_type::datetimeinfo, 488
indices
mo_common_variables::domain_meta, 492
inflowgauge
mo_mrm_global_variables, 238
inflowgaugeheadwater
mo_mrm_global_variables::domaininfo_mrm, 494
inflowgaugeidlist
mo_mrm_global_variables::domaininfo_mrm, 494
inflowgaugeindexlist
mo_mrm_global_variables::domaininfo_mrm, 494
inflowgaugenodelist
mo_mrm_global_variables::domaininfo_mrm, 494
init
mo_common_datetime_type::datetimeinfo, 488
mo_mrm_riv_temp_class, 329
mo_mrm_riv_temp_class::riv_temp_type, 529
init_area
mo_mrm_riv_temp_class, 329
mo_mrm_riv_temp_class::riv_temp_type, 529
init_domain_variable
mo_common_read_config, 109
init_domain_variable_for_master
mo_common_read_config, 110
init_eff_params
mo_mpr_startup, 224
init_indexarray_for_opti_data
mo_objective_function, 389
init_iter
mo_mrm_riv_temp_class, 330
mo_mrm_riv_temp_class::riv_temp_type, 530
init_lowres_level
mo_grid, 144
init_masked_zeros_l0
mo_mrm_river_head, 337
init_riv_temp
mo_mrm_riv_temp_class, 330
mo_mrm_riv_temp_class::riv_temp_type, 530
inputformat_gridded_lai
mo_mpr_global_variables, 192
inputformat_meteo_forcings
mo_global_variables, 132
intgrandfast
mo_neutrons, 383
iper_thres_runoff
mo_multi_param_reg, 370
is_new_day
mo_common_datetime_type::datetimeinfo, 489
is_new_month
mo_common_datetime_type::datetimeinfo, 489
is_new_year
mo_common_datetime_type::datetimeinfo, 490
is_present
mo_mpr_global_variables::soiltype, 539
is_read
mo_meteo_forcings, 153
is_start
mo_mrm_global_variables, 238
ismasterincomlocal
mo_common_variables::domain_meta, 492

istart
mo_common_variables::grid, 497
itest
mo_template, 462

jarvis_et_reduction
mo_soil_moisture, 451
julend
mo_common_variables::period, 519
julstart
mo_common_variables::period, 519

karman
mo_mpr_constants, 175
karstic_layer
mo_multi_param_reg, 371
ks
mo_mpr_global_variables::soiltype, 539
ks_c
mo_mpr_constants, 175

l0_asp
mo_mpr_global_variables, 192
l0_celerity
mo_mrm_global_variables, 238
l0_channel_depth
mo_mrm_global_variables, 239
l0_channel_elevation
mo_mrm_global_variables, 239
l0_check_input
mo_mpr_startup, 224
l0_check_input_routing
mo_mrm_init, 251
l0_coloutlet
mo_mrm_global_variables::domaininfo_mrm, 494
l0_domain
mo_common_variables, 120
l0_dracell
mo_mrm_global_variables, 239
l0_drasc
mo_mrm_global_variables, 239
l0_elev
mo_common_variables, 120
l0_facc
mo_mrm_global_variables, 239
l0_fdir
mo_mrm_global_variables, 239
l0_floodplain
mo_mrm_global_variables, 240
l0_fractionalcover_in_lx
mo_upscaling_operators, 468
l0_gaugeloc
mo_mrm_global_variables, 240
l0_geounit
mo_mpr_global_variables, 192
l0_grid_setup
mo_grid, 145
l0_gridded_lai
mo_mpr_global_variables, 192
l0_inflowgaugeloc
mo_mrm_global_variables, 240
l0_l11_remap
mo_mrm_global_variables, 240
l0_l1_remap
mo_common_variables, 120
l0_lcover
mo_common_variables, 121
l0_noutlet
mo_mrm_global_variables, 240
mo_mrm_global_variables::domaininfo_mrm, 494
l0_river_head_mon_sum
mo_mrm_global_variables, 241
l0_rowoutlet
mo_mrm_global_variables::domaininfo_mrm, 494
l0_slope
mo_mpr_global_variables, 193
mo_mrm_global_variables, 241
l0_slope_emp
mo_mpr_global_variables, 193
l0_soilid
mo_mpr_global_variables, 193
l0_streamnet
mo_mrm_global_variables, 241
l0_variable_init
mo_mpr_startup, 225
l0datafrom
mo_common_variables::domain_meta, 492
l11_afloodplain
mo_mrm_global_variables, 241
l11_air_temp
mo_mrm_riv_temp_class::riv_temp_type, 533
l11_areacell
mo_mrm_global_variables, 241
l11_bankfull_runoff_in
mo_mrm_global_variables, 241
l11_c1
mo_mrm_global_variables, 241
l11_c2
mo_mrm_global_variables, 242
l11_calc_celerity
mo_mrm_net_startup, 264
l11_celerity
mo_mrm_global_variables, 242
l11_cellcoor
mo_mrm_global_variables, 242
l11_colout
mo_mrm_global_variables, 242
l11_e_acc
mo_mrm_pre_routing, 309
l11_facc
mo_mrm_global_variables, 242
l11_fcol
mo_mrm_global_variables, 242
l11_fdir
mo_mrm_global_variables, 243
l11_flow_accumulation
mo_mrm_net_startup, 265

l11_flow_direction
mo_mrm_net_startup, 266
l11_fraction_sealed_floodplain
mo_mrm_net_startup, 268
l11_fromn
mo_mrm_global_variables, 243
l11_frow
mo_mrm_global_variables, 243
l11_k
mo_mrm_global_variables, 243
l11_l1_id
mo_mrm_global_variables, 243
l11_l1_mapping
mo_mrm_net_startup, 269
l11_label
mo_mrm_global_variables, 244
l11_length
mo_mrm_global_variables, 244
l11_link_location
mo_mrm_net_startup, 269
l11_linkin_facc
mo_mrm_global_variables, 244
l11_lrad_in
mo_mrm_riv_temp_class::riv_temp_type, 533
l11_meandering
mo_mrm_global_variables, 244
l11_meteo_acc
mo_mrm_pre_routing, 309
l11_netperm
mo_mrm_global_variables, 244
l11_nlinkfracfpimp
mo_mrm_global_variables, 244
l11_noutlets
mo_mrm_global_variables, 245
l11_qmod
mo_mrm_global_variables, 245
l11_qout
mo_mrm_global_variables, 245
l11_qtin
mo_mrm_global_variables, 245
l11_qtr
mo_mrm_global_variables, 245
l11_riv_areas
mo_mrm_riv_temp_class::riv_temp_type, 534
l11_riv_widths
mo_mrm_riv_temp_class::riv_temp_type, 534
l11_rorder
mo_mrm_global_variables, 245
l11_routing
mo_mrm_routing, 339
l11_routing_e
mo_mrm_riv_temp_class, 332
mo_mrm_riv_temp_class::riv_temp_type, 531
l11_routing_order
mo_mrm_net_startup, 270
l11_rowout
mo_mrm_global_variables, 246
l11_runoff_acc
mo_mrm_pre_routing, 310
l11_set_drain_outlet_gauges
mo_mrm_net_startup, 271
l11_set_network_topology
mo_mrm_net_startup, 272
l11_sink
mo_mrm_global_variables, 246
l11_slope
mo_mrm_global_variables, 246
l11_srad_net
mo_mrm_riv_temp_class::riv_temp_type, 534
l11_stream_features
mo_mrm_net_startup, 273
l11_tcol
mo_mrm_global_variables, 246
l11_ton
mo_mrm_global_variables, 246
l11_trow
mo_mrm_global_variables, 247
l11_tsrout
mo_mrm_global_variables, 247
l11_xi
mo_mrm_global_variables, 247
l1_absvappress
mo_global_variables, 132
l1_acc_ssrd
mo_mrm_riv_temp_class::riv_temp_type, 534
l1_acc_strd
mo_mrm_riv_temp_class::riv_temp_type, 534
l1_acc_temp
mo_mrm_riv_temp_class::riv_temp_type, 534
l1_aeroresist
mo_mpr_global_variables, 193
l1_aetcanopy
mo_global_variables, 132
l1_aetsealed
mo_global_variables, 132
l1_aetsoil
mo_global_variables, 132
l1_alpha
mo_mpr_global_variables, 193
l1_baseflow
mo_global_variables, 133
l1_degday
mo_mpr_global_variables, 193
l1_degdayinc
mo_mpr_global_variables, 194
l1_degdaymax
mo_mpr_global_variables, 194
l1_degdaynopre
mo_mpr_global_variables, 194
l1_etobs
mo_global_variables, 133
l1_fasp
mo_mpr_global_variables, 194
l1_fastrunoff
mo_global_variables, 133
l1_froots

mo_mpr_global_variables, 194
l1_fsealed
mo_mpr_global_variables, 195
l1_harsamcoeff
mo_mpr_global_variables, 195
l1_infilsoil
mo_global_variables, 133
l1_inter
mo_global_variables, 133
l1_jarvis_thresh_c1
mo_mpr_global_variables, 195
l1_karstloss
mo_mpr_global_variables, 195
l1_kbaseflow
mo_mpr_global_variables, 195
l1_kfastflow
mo_mpr_global_variables, 195
l1_kperco
mo_mpr_global_variables, 196
l1_kslowflow
mo_mpr_global_variables, 196
l1_l11_id
mo_mrm_global_variables, 247
l1_l11_remap
mo_mrm_global_variables, 247
l1_maxinter
mo_mpr_global_variables, 196
l1_melt
mo_global_variables, 134
l1_netrad
mo_global_variables, 134
l1_neutrons
mo_global_variables, 134
l1_neutronsdata
mo_global_variables, 134
l1_neutronsdata_mask
mo_global_variables, 134
l1_neutronsobs
mo_global_variables, 134
l1_percol
mo_global_variables, 134
l1_pet
mo_global_variables, 135
l1_pet_calc
mo_global_variables, 135
l1_pet_weights
mo_global_variables, 135
l1_petlaicorfactor
mo_mpr_global_variables, 196
l1_pre
mo_global_variables, 135
l1_pre_weights
mo_global_variables, 135
l1_preeffect
mo_global_variables, 135
l1_prietayalpha
mo_mpr_global_variables, 196
l1_rain
mo_global_variables, 136
l1_runoff_e
mo_mrm_riv_temp_class::riv_temp_type, 535
l1_runoffseal
mo_global_variables, 136
l1_satstw
mo_global_variables, 136
l1_sealedthresh
mo_mpr_global_variables, 196
l1_sealstw
mo_global_variables, 136
l1_slowrunoff
mo_global_variables, 136
l1_sm
mo_global_variables, 137
l1_sm_mask
mo_global_variables, 137
l1_smobs
mo_global_variables, 137
l1_snow
mo_global_variables, 137
l1_snowpack
mo_global_variables, 137
l1_soilmoist
mo_global_variables, 137
l1_soilmoistexp
mo_mpr_global_variables, 197
l1_soilmoistfc
mo_mpr_global_variables, 197
l1_soilmoistsat
mo_mpr_global_variables, 197
l1_ssrd
mo_global_variables, 138
l1_strd
mo_global_variables, 138
l1_surfresist
mo_mpr_global_variables, 197
l1_tann
mo_global_variables, 138
l1_temp
mo_global_variables, 138
l1_temp_weights
mo_global_variables, 138
l1_tempthresh
mo_mpr_global_variables, 197
l1_throughfall
mo_global_variables, 138
l1_tmax
mo_global_variables, 138
l1_tmin
mo_global_variables, 139
l1_total_runoff
mo_global_variables, 139
mo_runoff, 443
l1_total_runoff_in
mo_mrm_global_variables, 247
l1_twsaobs
mo_global_variables, 139

l1_unsatstw
mo_global_variables, 139
l1_unsatthresh
mo_mpr_global_variables, 198
l1_wiltingpoint
mo_mpr_global_variables, 198
l1_windspeed
mo_global_variables, 139
l2_variable_init
mo_startup, 457
lai_factor_surfresi
mo_mpr_constants, 176
lai_offset_surfresi
mo_mpr_constants, 176
laiboundaries
mo_mpr_global_variables, 198
lailut
mo_mpr_global_variables, 198
laiper
mo_mpr_global_variables, 198
laiunitlist
mo_mpr_global_variables, 198
laivarname
mo_common_constants, 86
landcoverperiodsvarname
mo_common_constants, 87
lc_year_end
mo_common_variables, 121
lc_year_start
mo_common_variables, 121
lcfilename
mo_common_variables, 121
lcyearid
mo_common_mhm_mrm_variables, 102
ld
mo_mpr_global_variables::soiltype, 539
left_bound
mo_common_variables::gridremapper, 499
level0
mo_common_variables, 121
level1
mo_common_variables, 122
level11
mo_mrm_global_variables, 248
level2
mo_global_variables, 139
limb_densities
mo_mrm_signatures, 347
loglikelihood_evin2013_2
mo_mrm_objective_function_runoff, 279
loglikelihood_stddev
mo_mrm_objective_function_runoff, 281
loglikelihood_trend_no_autocorr
mo_mrm_objective_function_runoff, 282
lookupintegral
mo_neutrons, 384
low_bnd_iter
mo_mrm_riv_temp_class::riv_temp_type, 535
low_res_grid
mo_common_variables::gridremapper, 499
lower_bound
mo_common_variables::gridremapper, 499
lowres_id_on_highres
mo_common_variables::gridremapper, 499

majority_statistics
mo_upscaling_operators, 469
mapcoordinates
mo_grid, 146
mask
mo_common_variables::grid, 497
mo_mrm_write_fluxes_states::outputvariable, 512
mo_write_fluxes_states::outputvariable, 516
max_iter
mo_mrm_riv_temp_class::riv_temp_type, 535
max_surfresist
mo_mpr_constants, 176
maxgeounit
mo_mpr_constants, 176
maximummonthlyflow
mo_mrm_signatures, 348
maxnlcovers
mo_common_constants, 87
maxnodomains
mo_common_constants, 87
maxnogauges
mo_mrm_constants, 228
maxnosoilhorizons
mo_mpr_constants, 176
mcmc_error_params
mo_common_mhm_mrm_variables, 102
mcmc_opti
mo_common_mhm_mrm_variables, 102
mean_dp
mo_template, 461
mo_template::mean, 501
mean_sp
mo_template, 461
mo_template::mean, 501
mend
mo_common_variables::period, 519
meteo_forcings_wrapper
mo_meteo_forcings, 153
meteo_weights_wrapper
mo_meteo_forcings, 154
mhm
mo_mhm, 158
mhm_details
mo_common_variables, 122
mhm_driver
mhm_driver.f90, 559
mhm_driver.f90
mhm_driver, 559
mhm_eval
mo_mhm_eval, 168
mhm_initialize
mo_startup, 458

mhm_read_config
mo_mhm_read_config, 172
mhmfilerestartin
mo_common_mhm_mrm_variables, 102
mhmfilerestartout
mo_common_variables, 122
mo_canopy_interc, 83
canopy_interc, 83
mo_check, 84
check_dir, 84
mo_common_constants, 85
eps_dp, 86
eps_sp, 86
laivarname, 86
landcoverperiodsvarname, 87
maxnlcovers, 87
maxnodomains, 87
ncolpars, 87
nodata_dp, 87
nodata_i4, 88
p1_initstatefluxes, 88
soilhorizonsvarname, 88
mo_common_datetime_type, 89
datetimeinfo_increment, 89
datetimeinfo_init, 89
datetimeinfo_update_lai_timestep, 89
datetimeinfo_writeout, 89
mo_common_datetime_type::datetimeinfo, 488
day, 489
hour, 489
ilai, 489
increment, 488
init, 488
is_new_day, 489
is_new_month, 489
is_new_year, 490
month, 490
newtime, 490
ntimesteps, 490
prev_day, 490
prev_month, 490
prev_year, 490
tindex_out, 491
update_lai_timestep, 489
writeout, 489
year, 491
yid, 491
mo_common_file, 90
file_config, 90
file_dem, 90
uconfig, 91
udem, 91
ulcoverclass, 91
mo_common_functions, 91
in_bound, 92
mo_common_mhm_mrm_file, 92
file_opti, 93
file_opti_nml, 93
uopti, 93
uopti_nml, 93
mo_common_mhm_mrm_mpi_tools, 94
distribute_parameterset, 94
get_parameterset, 95
mo_common_mhm_mrm_read_config, 95
check_optimization_settings, 96
common_check_resolution, 96
common_mhm_mrm_read_config, 97
period_copy_period_data, 98
mo_common_mhm_mrm_restart, 99
check_consistency_element_dp, 99
check_consistency_element_i4, 100
check_dimension_consistency, 100
mo_common_mhm_mrm_restart::check_consistency_element, 487
check_consistency_element_dp, 487
check_consistency_element_i4, 487
mo_common_mhm_mrm_variables, 100
c2tstu, 101
dds_r, 101
evalper, 102
lcyearid, 102
mcmc_error_params, 102
mcmc_opti, 102
mhmfilerestartin, 102
mrm_coupling_mode, 103
mrm_read_river_network, 103
mrmfilerestartin, 103
nerror_model, 103
niterations, 103
ntstepday, 103
opti_function, 104
opti_method, 104
optimize, 104
optimize_restart, 104
read_restart, 104
readper, 105
resolutionrouting, 105
sa_temp, 105
sce_ngs, 105
sce_npg, 105
sce_nps, 106
seed, 106
simper, 106
timestep, 106
warmingdays, 106
warmper, 107
mo_common_read_config, 107
common_read_config, 107
distribute_processes_to_domains_according_to_role, 109
distributedomainsroundrobin, 109
init_domain_variable, 109
init_domain_variable_for_master, 110
set_land_cover_scenes_id, 110
mo_common_read_data, 111
read_dem, 112

read_lcover, 113
mo_common_restart, 113
read_grid_info, 114
write_grid_info, 115
mo_common_variables, 116
alma_convention, 117
comm, 117
contact, 117
conventions, 118
dircommonfiles, 118
dirconfigout, 118
dirlcover, 118
dirmorpho, 118
dirout, 118
domainmeta, 119
filelatlon, 119
global_parameters, 119
global_parameters_name, 120
history, 120
iflag_cordinate_sys, 120
l0_domain, 120
l0_elev, 120
l0_l1_remap, 120
l0_lcover, 121
lc_year_end, 121
lc_year_start, 121
lcfilename, 121
level0, 121
level1, 122
mhm_details, 122
mhmfilerestartout, 122
mrmfilerestartout, 122
nlcoverscene, 123
nprocesses, 123
nuniquel0domains, 123
processmatrix, 123
project_details, 123
resolutionhydrology, 124
setup_description, 124
simulation_type, 124
write_restart, 124
mo_common_variables::domain_meta, 491
comlocal, 491
commaster, 492
dorouting, 492
indices, 492
ismasterincomlocal, 492
l0datafrom, 492
ndomains, 492
optidata, 492
overallnumberofdomains, 493
mo_common_variables::grid, 496
cellarea, 496
cellcoor, 497
cellsize, 497
id, 497
iend, 497
istart, 497
mask, 497
ncells, 497
ncols, 498
nodata_value, 498
nrows, 498
x, 498
xllcorner, 498
y, 498
yllcorner, 498
mo_common_variables::gridremapper, 499
high_res_grid, 499
left_bound, 499
low_res_grid, 499
lower_bound, 499
lowres_id_on_highres, 499
n_subcells, 500
right_bound, 500
upper_bound, 500
mo_common_variables::period, 518
dend, 519
dstart, 519
julend, 519
julstart, 519
mend, 519
mstart, 519
nobs, 520
yend, 520
ystart, 520
mo_file, 124
file_defoutput, 125
file_main, 125
file_namelist_mhm, 125
file_namelist_mhm_param, 126
udefoutput, 126
unamelist_mhm, 126
unamelist_mhm_param, 126
version, 126
version_date, 126
mo_global_variables, 127
dirabsvappressure, 128
dirmaxtemperature, 129
dirmintemperature, 129
dirnetradiation, 129
dirprecipitation, 129
dirradiation, 129
dirreferenceet, 130
dirtemperature, 130
dirwindspeed, 130
evap_coeff, 130
fday_pet, 130
fday_prec, 130
fday_ssrd, 131
fday_strd, 131
fday_temp, 131
fnight_pet, 131
fnight_prec, 131
fnight_ssrd, 131
fnight_strd, 131

fnight_temp, 132
inputformat_meteo_forcings, 132
l1_absvappress, 132
l1_aetcanopy, 132
l1_aetsealed, 132
l1_aetsoil, 132
l1_baseflow, 133
l1_etobs, 133
l1_fastrunoff, 133
l1_infilsoil, 133
l1_inter, 133
l1_melt, 134
l1_netrad, 134
l1_neutrons, 134
l1_neutronsdata, 134
l1_neutronsdata_mask, 134
l1_neutronsobs, 134
l1_percol, 134
l1_pet, 135
l1_pet_calc, 135
l1_pet_weights, 135
l1_pre, 135
l1_pre_weights, 135
l1_preeffect, 135
l1_rain, 136
l1_runoffseal, 136
l1_satstw, 136
l1_sealstw, 136
l1_slowrunoff, 136
l1_sm, 137
l1_sm_mask, 137
l1_smobs, 137
l1_snow, 137
l1_snowpack, 137
l1_soilmoist, 137
l1_ssrd, 138
l1_strd, 138
l1_tann, 138
l1_temp, 138
l1_temp_weights, 138
l1_throughfall, 138
l1_tmax, 138
l1_tmin, 139
l1_total_runoff, 139
l1_twsaobs, 139
l1_unsatstw, 139
l1_windspeed, 139
level2, 139
neutron_integral_afast, 140
nsoilhorizons_sm_input, 140
output_deflate_level, 140
output_double_precision, 140
outputflxstate, 140
read_meteo_weights, 140
routingstates, 141
timestep_model_inputs, 141
timestep_model_outputs, 141
mo_grid, 141
calculate_grid_properties, 142
geocoordinates, 143
init_lowres_level, 144
l0_grid_setup, 145
mapcoordinates, 146
set_domain_indices, 147
mo_init_states, 148
variables_alloc, 148
variables_default_init, 149
mo_meteo_forcings, 150
chunk_config, 151
chunk_size, 152
is_read, 153
meteo_forcings_wrapper, 153
meteo_weights_wrapper, 154
prepare_meteo_forcings_data, 156
mo_mhm, 157
mhm, 158
mo_mhm_constants, 163
c1_initstatesm, 164
cosmic_alpha, 164
cosmic_bd, 164
cosmic_l1, 164
cosmic_l2, 164
cosmic_l3, 164
cosmic_l4, 164
cosmic_n, 165
cosmic_vwclat, 165
desilets_a0, 165
desilets_a1, 165
desilets_a2, 165
duffiedelta1, 165
duffiedelta2, 166
duffiedr, 166
h2odens, 166
harsamconst, 166
noutflxstate, 166
p2_initstatefluxes, 166
p3_initstatefluxes, 167
p4_initstatefluxes, 167
p5_initstatefluxes, 167
satpressureslope1, 167
tetens_c1, 167
tetens_c2, 167
tetens_c3, 168
mo_mhm_eval, 168
mhm_eval, 168
mo_mhm_read_config, 171
mhm_read_config, 172
mo_mpr_constants, 173
bulkdens_orgmatter, 175
c1_initstatesm, 175
field_cap_c1, 175
field_cap_c2, 175
karman, 175
ks_c, 175
lai_factor_surfresi, 176
lai_offset_surfresi, 176

max_surfresist, 176
maxgeounit, 176
maxnosoilhorizons, 176
nlcover_class, 176
p2_initstatefluxes, 177
p3_initstatefluxes, 177
p4_initstatefluxes, 177
p5_initstatefluxes, 177
pwp_c, 177
pwp_matpot_thetar, 177
vgenuchten_sandtresh, 177
vgenuchtenn_c1, 178
vgenuchtenn_c10, 178
vgenuchtenn_c11, 178
vgenuchtenn_c12, 178
vgenuchtenn_c13, 178
vgenuchtenn_c14, 178
vgenuchtenn_c15, 179
vgenuchtenn_c16, 179
vgenuchtenn_c17, 179
vgenuchtenn_c18, 179
vgenuchtenn_c2, 179
vgenuchtenn_c3, 179
vgenuchtenn_c4, 180
vgenuchtenn_c5, 180
vgenuchtenn_c6, 180
vgenuchtenn_c7, 180
vgenuchtenn_c8, 180
vgenuchtenn_c9, 180
windmeasheight, 180
mo_mpr_eval, 181
mpr_eval, 181
mo_mpr_file, 183
file_aspect, 184
file_geolut, 184
file_hydrogeoclass, 185
file_laiclass, 185
file_lailut, 185
file_main, 185
file_meteo_binary_end, 185
file_meteo_header, 186
file_namelist_mpr, 186
file_namelist_mpr_param, 186
file_slope, 186
file_soil_database, 186
file_soil_database_1, 186
file_soilclass, 187
uaspect, 187
ugeolut, 187
uhydrogeoclass, 187
ulaiclass, 187
ulailut, 187
umeteo, 188
umeteo_header, 188
unamelist_mpr, 188
unamelist_mpr_param, 188
uslope, 188
usoil_database, 189
usoilclass, 189
version, 189
version_date, 189
mo_mpr_global_variables, 189
dirgridded_lai, 191
fracsealed_cityarea, 191
geounitkar, 191
geounitlist, 191
horizondepth_mhm, 191
iflag_soildb, 192
inputformat_gridded_lai, 192
l0_asp, 192
l0_geounit, 192
l0_gridded_lai, 192
l0_slope, 193
l0_slope_emp, 193
l0_soilid, 193
l1_aeroresist, 193
l1_alpha, 193
l1_degday, 193
l1_degdayinc, 194
l1_degdaymax, 194
l1_degdaynopre, 194
l1_fasp, 194
l1_froots, 194
l1_fsealed, 195
l1_harsamcoeff, 195
l1_jarvis_thresh_c1, 195
l1_karstloss, 195
l1_kbaseflow, 195
l1_kfastflow, 195
l1_kperco, 196
l1_kslowflow, 196
l1_maxinter, 196
l1_petlaicorfactor, 196
l1_prietayalpha, 196
l1_sealedthresh, 196
l1_soilmoistexp, 197
l1_soilmoistfc, 197
l1_soilmoistsat, 197
l1_surfresist, 197
l1_tempthresh, 197
l1_unsatthresh, 198
l1_wiltingpoint, 198
laiboundaries, 198
lailut, 198
laiper, 198
laiunitlist, 198
ngeounits, 198
nlai, 199
nlaiclass, 199
nsoilhorizons_mhm, 199
nsoiltypes, 199
soildb, 199
tillagedepth, 200
timestep_lai_input, 200
mo_mpr_global_variables::soiltype, 538
clay, 538

db, 538
dbm, 538
depth, 538
id, 539
is_present, 539
ks, 539
ld, 539
nhorizons, 539
ntillhorizons, 539
rzdepth, 539
sand, 540
thetafc, 540
thetafc_till, 540
thetapw, 540
thetapw_till, 540
thetas, 540
thetas_till, 540
ud, 540
wd, 541
mo_mpr_pet, 200
bulksurface_resistance, 201
pet_correctbyasp, 202
pet_correctbylai, 203
priestley_taylor_alpha, 204
mo_mpr_read_config, 206
mpr_read_config, 206
mo_mpr_restart, 207
unpack_field_and_write_1d_dp, 208
unpack_field_and_write_1d_i4, 208
unpack_field_and_write_2d_dp, 209
unpack_field_and_write_3d_dp, 209
write_eff_params, 209
write_mpr_restart_files, 210
mo_mpr_restart::unpack_field_and_write, 543
unpack_field_and_write_1d_dp, 544
unpack_field_and_write_1d_i4, 544
unpack_field_and_write_2d_dp, 545
unpack_field_and_write_3d_dp, 545
mo_mpr_runoff, 211
mpr_runoff, 212
mo_mpr_smhorizons, 214
mpr_smhorizons, 214
mo_mpr_soilmoist, 216
field_cap, 217
genuchten, 218
hydro_cond, 219
mpr_sm, 220
pwp, 222
mo_mpr_startup, 223
init_eff_params, 224
l0_check_input, 224
l0_variable_init, 225
mpr_initialize, 226
mo_mrm_constants, 228
deltah, 228
given_ts, 228
maxnogauges, 228
noutflxstate, 229
nroutingstates, 229
rout_space_weight, 229
mo_mrm_file, 229
file_config, 230
file_daily_discharge, 231
file_defoutput, 231
file_facc, 231
file_fdir, 231
file_gaugeloc, 231
file_gw_output, 232
file_main, 232
file_mrm_output, 232
file_namelist_mrm, 232
file_namelist_param_mrm, 232
file_slope, 232
ncfile_discharge, 233
uconfig, 233
udaily_discharge, 233
udefoutput, 233
udischarge, 233
ufacc, 234
ufdir, 234
ugaugeloc, 234
unamelist_mrm, 234
unamelist_param_mrm, 234
uslope, 234
version, 235
version_date, 235
mo_mrm_global_variables, 235
dirbankfullrunoff, 237
dirgauges, 237
dirtotalrunoff, 237
domain_mrm, 237
filenametotalrunoff, 238
gauge, 238
gw_coupling, 238
inflowgauge, 238
is_start, 238
l0_celerity, 238
l0_channel_depth, 239
l0_channel_elevation, 239
l0_dracell, 239
l0_drasc, 239
l0_facc, 239
l0_fdir, 239
l0_floodplain, 240
l0_gaugeloc, 240
l0_inflowgaugeloc, 240
l0_l11_remap, 240
l0_noutlet, 240
l0_river_head_mon_sum, 241
l0_slope, 241
l0_streamnet, 241
l11_afloodplain, 241
l11_areacell, 241
l11_bankfull_runoff_in, 241
l11_c1, 241
l11_c2, 242

l11_celerity, 242
l11_cellcoor, 242
l11_colout, 242
l11_facc, 242
l11_fcol, 242
l11_fdir, 243
l11_fromn, 243
l11_frow, 243
l11_k, 243
l11_l1_id, 243
l11_label, 244
l11_length, 244
l11_linkin_facc, 244
l11_meandering, 244
l11_netperm, 244
l11_nlinkfracfpimp, 244
l11_noutlets, 245
l11_qmod, 245
l11_qout, 245
l11_qtin, 245
l11_qtr, 245
l11_rorder, 245
l11_rowout, 246
l11_sink, 246
l11_slope, 246
l11_tcol, 246
l11_ton, 246
l11_trow, 247
l11_tsrout, 247
l11_xi, 247
l1_l11_id, 247
l1_l11_remap, 247
l1_total_runoff_in, 247
level11, 248
mrm_runoff, 248
ngaugeslocal, 248
ngaugestotal, 248
ninflowgaugestotal, 248
nmeasperday, 249
ntstepday, 249
output_deflate_level_mrm, 249
output_double_precision_mrm, 249
outputflxstate_mrm, 249
riv_temp_pcs, 249
timestep_model_outputs_mrm, 250
varnametotalrunoff, 250
mo_mrm_global_variables::domaininfo_mrm, 493
gaugeidlist, 493
gaugeindexlist, 493
gaugenodelist, 493
inflowgaugeheadwater, 494
inflowgaugeidlist, 494
inflowgaugeindexlist, 494
inflowgaugenodelist, 494
l0_coloutlet, 494
l0_noutlet, 494
l0_rowoutlet, 494
ngauges, 495
ninflowgauges, 495
mo_mrm_global_variables::gaugingstation, 495
domainid, 495
fname, 495
gaugeid, 495
q, 496
t, 496
mo_mrm_init, 250
config_output, 251
l0_check_input_routing, 251
mrm_configuration, 252
mrm_init, 253
print_startup_message, 256
variables_alloc_routing, 256
variables_default_init_routing, 257
mo_mrm_mpr, 258
mrm_init_param, 258
mrm_update_param, 259
reg_rout, 260
mo_mrm_net_startup, 261
celllength, 262
get_distance_two_lat_lon_points, 264
l11_calc_celerity, 264
l11_flow_accumulation, 265
l11_flow_direction, 266
l11_fraction_sealed_floodplain, 268
l11_l1_mapping, 269
l11_link_location, 269
l11_routing_order, 270
l11_set_drain_outlet_gauges, 271
l11_set_network_topology, 272
l11_stream_features, 273
movedownonecell, 274
moveup, 275
mo_mrm_net_startup.f90
calculate_l11_flow_accumulation, 587
mo_mrm_objective_function_runoff, 276
extract_runoff, 278
loglikelihood_evin2013_2, 279
loglikelihood_stddev, 281
loglikelihood_trend_no_autocorr, 282
multi_objective_ae_fdc_lsv_nse_djf, 283
multi_objective_lnnse_highflow_lnnse_lowflow, 285
multi_objective_lnnse_highflow_lnnse_lowflow_2, 286
multi_objective_nse_lnnse, 287
multi_objective_runoff, 288
objective_equal_nse_lnnse, 289
objective_kge, 291
objective_lnnse, 292
objective_multiple_gauges_kge_power6, 293
objective_nse, 295
objective_power6_nse_lnnse, 296
objective_sse, 297
objective_sse_boxcox, 298
objective_weighted_nse, 300
parameter_regularization, 301

single_objective_runoff, 302
single_objective_runoff_master, 303
single_objective_runoff_subprocess, 304
mo_mrm_pre_routing, 306
add_inflow, 307
calc_l1_runoff_e, 308
l11_e_acc, 309
l11_meteo_acc, 309
l11_runoff_acc, 310
mo_mrm_read_config, 311
mrm_read_config, 312
read_mrm_routing_params, 313
mo_mrm_read_data, 314
mrm_read_bankfull_runoff, 315
mrm_read_discharge, 315
mrm_read_l0_data, 316
mrm_read_total_runoff, 318
rotate_fdir_variable, 318
mo_mrm_restart, 319
mrm_read_restart_config, 320
mrm_read_restart_states, 320
mrm_write_restart, 321
mo_mrm_riv_temp_class, 323
acc_source_e, 324
alloc_lateral, 325
config, 326
dealloc_lateral, 326
finalize_source_e, 326
get_e_io, 327
get_lat_heat, 328
get_lrad_out, 328
get_sens_heat, 329
init, 329
init_area, 329
init_iter, 330
init_riv_temp, 330
l11_routing_e, 332
next_iter, 333
reset_timestep, 333
mo_mrm_riv_temp_class::riv_temp_type, 523
acc_source_e, 525
active, 532
albedo_water, 532
alloc_lateral, 526
bisect_iter, 532
case, 532
config, 526
dealloc_lateral, 527
delta_iter, 532
delta_t, 533
dir_riv_widths, 533
e11, 533
emissivity_water, 533
finalize_source_e, 527
first_iter, 533
get_e_io, 527
get_lat_heat, 528
get_lrad_out, 528
get_sens_heat, 529
init, 529
init_area, 529
init_iter, 530
init_riv_temp, 530
l11_air_temp, 533
l11_lrad_in, 533
l11_riv_areas, 534
l11_riv_widths, 534
l11_routing_e, 531
l11_srad_net, 534
l1_acc_ssrd, 534
l1_acc_strd, 534
l1_acc_temp, 534
l1_runoff_e, 535
low_bnd_iter, 535
max_iter, 535
netnode_e_in, 535
netnode_e_mod, 535
netnode_e_out, 535
netnode_e_r, 536
next_iter, 531
nml_name, 536
pt_a_water, 536
reset_timestep, 531
riv_widths_file, 536
riv_widths_name, 536
river_temp, 536
s11, 537
step_iter, 537
ts_cnt, 537
turb_heat_ex_coeff, 537
up_bnd_iter, 537
up_iter, 537
mo_mrm_river_head, 333
avg_and_write_timestep, 333
calc_channel_elevation, 334
calc_river_head, 335
calc_slope, 335
create_output, 336
init_masked_zeros_l0, 337
nc_riverhead, 338
nc_time, 338
reset_sum, 337
sum_counter, 338
time_counter, 338
mo_mrm_routing, 338
l11_routing, 339
mrm_routing, 340
mo_mrm_signatures, 343
autocorrelation, 344
flowdurationcurve, 345
limb_densities, 347
maximummonthlyflow, 348
moments, 348
peakdistribution, 349
runoffratio, 350
zeroflowratio, 351

mo_mrm_write, 352
mrm_write, 353
mrm_write_optifile, 354
mrm_write_optinamelist, 354
mrm_write_output_fluxes, 355
nc, 358
write_configfile, 356
write_daily_obs_sim_discharge, 357
mo_mrm_write_fluxes_states, 358
close, 359
createoutputfile, 359
newoutputdataset, 360
newoutputvariable, 361
updatedataset, 362
updatevariable, 363
writetimestep, 364
writevariableattributes, 364
writevariabletimestep, 365
mo_mrm_write_fluxes_states::outputdataset, 502
all, 503
close, 502
count, 503
counter, 503
created, 503
domain, 503
id, 504
idomain, 504
nc, 504
ncdataset, 504
steps, 504
store, 504
time, 504
to, 505
updatedataset, 503
variables, 505
vars, 505
write, 505
writetimestep, 503
written, 505
mo_mrm_write_fluxes_states::outputvariable, 509
average, 510
avg, 510
before, 510
between, 511
calls, 511
contains, 511
count, 511
counter, 511
data, 511
mask, 512
nc, 512
ncdataset, 512
number, 512
of, 512
reconstruct, 512
store, 512
the, 512, 513
to, 513
updatevariable, 510, 513
variable, 513
which, 513
writes, 513
writevariabletimestep, 510
writing, 514
mo_multi_param_reg, 366
aerodynamical_resistance, 367
baseflow_param, 368
canopy_intercept_param, 369
iper_thres_runoff, 370
karstic_layer, 371
mpr, 372
snow_acc_melt_param, 376
mo_neutrons, 378
approx_mon_int, 378
approx_mon_int_eps, 379
approx_mon_int_steps, 380
cosmic, 381
desiletsn0, 382
intgrandfast, 383
lookupintegral, 384
oldintegration, 384
tabularintegralafast, 385
mo_objective_function, 386
convert_tws_to_twsa, 387
create_domain_avg_et, 388
create_domain_avg_tws, 388
init_indexarray_for_opti_data, 389
objective, 390
objective_et_kge_catchment_avg, 391
objective_kge_q_et, 393
objective_kge_q_rmse_et, 395
objective_kge_q_rmse_tws, 396
objective_kge_q_sm_corr, 397
objective_master, 398
objective_neutrons_kge_catchment_avg, 399
objective_q_et_tws_kge_catchment_avg, 400
objective_sm_corr, 402
objective_sm_kge_catchment_avg, 403
objective_sm_pd, 405
objective_sm_sse_standard_score, 406
objective_subprocess, 407
mo_optimization, 409
optimization, 409
mo_pet, 410
extraterr_rad_approx, 411
pet_hargreaves, 412
pet_penman, 413
pet_priestly, 415
sat_vap_pressure, 416
slope_satpressure, 416
mo_prepare_gridded_lai, 417
prepare_gridded_daily_lai_data, 418
prepare_gridded_mean_monthly_lai_data, 419
mo_read_latlon, 420
read_latlon, 420
mo_read_lut, 421

read_geoformation_lut, 422
read_lai_lut, 423
mo_read_nc, 424
get_time_vector_and_select, 425
read_const_nc, 425
read_nc, 426
read_weights_nc, 428
mo_read_optional_data, 429
readoptidataobs, 430
mo_read_spatial_data, 431
read_header_ascii, 431
read_spatial_data_ascii_dp, 432
read_spatial_data_ascii_i4, 433
mo_read_spatial_data::read_spatial_data_ascii, 520
read_spatial_data_ascii_dp, 521
read_spatial_data_ascii_i4, 522
mo_read_timeseries, 434
read_timeseries, 434
mo_read_wrapper, 435
check_consistency_lut_map, 436
read_data, 437
mo_restart, 438
read_restart_states, 439
unpack_field_and_write_1d_dp, 440
unpack_field_and_write_1d_i4, 441
unpack_field_and_write_2d_dp, 441
unpack_field_and_write_3d_dp, 441
write_restart_files, 441
mo_restart::unpack_field_and_write, 545
unpack_field_and_write_1d_dp, 546
unpack_field_and_write_1d_i4, 546
unpack_field_and_write_2d_dp, 546
unpack_field_and_write_3d_dp, 547
mo_runoff, 443
l1_total_runoff, 443
runoff_sat_zone, 444
runoff_unsat_zone, 445
mo_snow_accum_melt, 446
snow_accum_melt, 447
mo_soil_database, 448
generate_soil_database, 448
read_soil_lut, 449
mo_soil_moisture, 450
feddes_et_reduction, 451
jarvis_et_reduction, 451
soil_moisture, 452
mo_spatial_agg_disagg_forcing, 454
spatial_aggregation_3d, 455
spatial_aggregation_4d, 455
spatial_disaggregation_3d, 456
spatial_disaggregation_4d, 456
mo_spatial_agg_disagg_forcing::spatial_aggregation, 541
spatial_aggregation_3d, 541
spatial_aggregation_4d, 542
mo_spatial_agg_disagg_forcing::spatial_disaggregation, 542
spatial_disaggregation_3d, 543
spatial_disaggregation_4d, 543
mo_startup, 456
constants_init, 457
l2_variable_init, 457
mhm_initialize, 458
mo_template, 460
circum, 460
itest, 462
mean_dp, 461
mean_sp, 461
pi_dp, 462
pi_sp, 462
mo_template::mean, 500
mean_dp, 501
mean_sp, 501
mo_temporal_disagg_forcing, 462
temporal_disagg_flux_daynight, 463
temporal_disagg_forcing, 464
temporal_disagg_meteo_weights, 466
temporal_disagg_state_daynight, 466
mo_upscaling_operators, 467
l0_fractionalcover_in_lx, 468
majority_statistics, 469
upscale_arithmetic_mean, 470
upscale_geometric_mean, 471
upscale_harmonic_mean, 472
upscale_p_norm, 473
mo_write_ascii, 474
write_configfile, 474
write_optifile, 475
write_optinamelist, 476
mo_write_fluxes_states, 477
close, 478
createoutputfile, 478
fluxesunit, 479
newoutputdataset, 480
newoutputvariable, 481
updatedataset, 482
updatevariable, 483
writetimestep, 484
writevariableattributes, 485
writevariabletimestep, 486
mo_write_fluxes_states::outputdataset, 505
all, 507
close, 506
count, 507
counter, 507
created, 507
domain, 507
id, 507
idomain, 507
nc, 508
ncdataset, 508
steps, 508
store, 508
time, 508
to, 508
updatedataset, 506

variables, 508
vars, 509
write, 509
writetimestep, 506
written, 509
mo_write_fluxes_states::outputvariable, 514
average, 515
avg, 515
before, 515
between, 515
calls, 515
contains, 516
count, 516
counter, 516
data, 516
mask, 516
nc, 516
ncdataset, 516
number, 517
of, 517
reconstruct, 517
store, 517
the, 517
to, 518
updatevariable, 515, 518
variable, 518
which, 518
writes, 518
writevariabletimestep, 515
writing, 518
moments
mo_mrm_signatures, 348
month
mo_common_datetime_type::datetimeinfo, 490
movedownonecell
mo_mrm_net_startup, 274
moveup
mo_mrm_net_startup, 275
mpr
mo_multi_param_reg, 372
mpr_driver
mpr_driver.f90, 580
mpr_driver.f90
mpr_driver, 580
mpr_eval
mo_mpr_eval, 181
mpr_initialize
mo_mpr_startup, 226
mpr_read_config
mo_mpr_read_config, 206
mpr_runoff
mo_mpr_runoff, 212
mpr_sm
mo_mpr_soilmoist, 220
mpr_smhorizons
mo_mpr_smhorizons, 214
mrm_configuration
mo_mrm_init, 252
mrm_coupling_mode
mo_common_mhm_mrm_variables, 103
mrm_init
mo_mrm_init, 253
mrm_init_param
mo_mrm_mpr, 258
mrm_read_bankfull_runoff
mo_mrm_read_data, 315
mrm_read_config
mo_mrm_read_config, 312
mrm_read_discharge
mo_mrm_read_data, 315
mrm_read_l0_data
mo_mrm_read_data, 316
mrm_read_restart_config
mo_mrm_restart, 320
mrm_read_restart_states
mo_mrm_restart, 320
mrm_read_river_network
mo_common_mhm_mrm_variables, 103
mrm_read_total_runoff
mo_mrm_read_data, 318
mrm_routing
mo_mrm_routing, 340
mrm_runoff
mo_mrm_global_variables, 248
mrm_update_param
mo_mrm_mpr, 259
mrm_write
mo_mrm_write, 353
mrm_write_optifile
mo_mrm_write, 354
mrm_write_optinamelist
mo_mrm_write, 354
mrm_write_output_fluxes
mo_mrm_write, 355
mrm_write_restart
mo_mrm_restart, 321
mrmfilerestartin
mo_common_mhm_mrm_variables, 103
mrmfilerestartout
mo_common_variables, 122
mstart
mo_common_variables::period, 519
multi_objective_ae_fdc_lsv_nse_djf
mo_mrm_objective_function_runoff, 283
multi_objective_lnnse_highflow_lnnse_lowflow
mo_mrm_objective_function_runoff, 285
multi_objective_lnnse_highflow_lnnse_lowflow_2
mo_mrm_objective_function_runoff, 286
multi_objective_nse_lnnse
mo_mrm_objective_function_runoff, 287
multi_objective_runoff
mo_mrm_objective_function_runoff, 288

n_subcells
mo_common_variables::gridremapper, 500
nc
mo_mrm_write, 358

mo_mrm_write_fluxes_states::outputdataset, 504
mo_mrm_write_fluxes_states::outputvariable, 512
mo_write_fluxes_states::outputdataset, 508
mo_write_fluxes_states::outputvariable, 516
nc_riverhead
mo_mrm_river_head, 338
nc_time
mo_mrm_river_head, 338
ncdataset
mo_mrm_write_fluxes_states::outputdataset, 504
mo_mrm_write_fluxes_states::outputvariable, 512
mo_write_fluxes_states::outputdataset, 508
mo_write_fluxes_states::outputvariable, 516
ncells
mo_common_variables::grid, 497
ncfile_discharge
mo_mrm_file, 233
ncolpars
mo_common_constants, 87
ncols
mo_common_variables::grid, 498
ndomains
mo_common_variables::domain_meta, 492
nerror_model
mo_common_mhm_mrm_variables, 103
netnode_e_in
mo_mrm_riv_temp_class::riv_temp_type, 535
netnode_e_mod
mo_mrm_riv_temp_class::riv_temp_type, 535
netnode_e_out
mo_mrm_riv_temp_class::riv_temp_type, 535
netnode_e_r
mo_mrm_riv_temp_class::riv_temp_type, 536
neutron_integral_afast
mo_global_variables, 140
newoutputdataset
mo_mrm_write_fluxes_states, 360
mo_write_fluxes_states, 480
newoutputvariable
mo_mrm_write_fluxes_states, 361
mo_write_fluxes_states, 481
newtime
mo_common_datetime_type::datetimeinfo, 490
next_iter
mo_mrm_riv_temp_class, 333
mo_mrm_riv_temp_class::riv_temp_type, 531
ngauges
mo_mrm_global_variables::domaininfo_mrm, 495
ngaugeslocal
mo_mrm_global_variables, 248
ngaugestotal
mo_mrm_global_variables, 248
ngeounits
mo_mpr_global_variables, 198
nhorizons
mo_mpr_global_variables::soiltype, 539
ninflowgauges
mo_mrm_global_variables::domaininfo_mrm, 495
ninflowgaugestotal
mo_mrm_global_variables, 248
niterations
mo_common_mhm_mrm_variables, 103
nlai
mo_mpr_global_variables, 199
nlaiclass
mo_mpr_global_variables, 199
nlcover_class
mo_mpr_constants, 176
nlcoverscene
mo_common_variables, 123
nmeasperday
mo_mrm_global_variables, 249
nml_name
mo_mrm_riv_temp_class::riv_temp_type, 536
nobs
mo_common_variables::period, 520
nodata_dp
mo_common_constants, 87
nodata_i4
mo_common_constants, 88
nodata_value
mo_common_variables::grid, 498
noutflxstate
mo_mhm_constants, 166
mo_mrm_constants, 229
nprocesses
mo_common_variables, 123
nroutingstates
mo_mrm_constants, 229
nrows
mo_common_variables::grid, 498
nsoilhorizons_mhm
mo_mpr_global_variables, 199
nsoilhorizons_sm_input
mo_global_variables, 140
nsoiltypes
mo_mpr_global_variables, 199
ntillhorizons
mo_mpr_global_variables::soiltype, 539
ntimesteps
mo_common_datetime_type::datetimeinfo, 490
ntstepday
mo_common_mhm_mrm_variables, 103
mo_mrm_global_variables, 249
number
mo_mrm_write_fluxes_states::outputvariable, 512
mo_write_fluxes_states::outputvariable, 517
nuniquel0domains
mo_common_variables, 123

objective
mo_objective_function, 390
objective_equal_nse_lnnse
mo_mrm_objective_function_runoff, 289
objective_et_kge_catchment_avg
mo_objective_function, 391
objective_kge

mo_mrm_objective_function_runoff, 291
objective_kge_q_et
mo_objective_function, 393
objective_kge_q_rmse_et
mo_objective_function, 395
objective_kge_q_rmse_tws
mo_objective_function, 396
objective_kge_q_sm_corr
mo_objective_function, 397
objective_lnnse
mo_mrm_objective_function_runoff, 292
objective_master
mo_objective_function, 398
objective_multiple_gauges_kge_power6
mo_mrm_objective_function_runoff, 293
objective_neutrons_kge_catchment_avg
mo_objective_function, 399
objective_nse
mo_mrm_objective_function_runoff, 295
objective_power6_nse_lnnse
mo_mrm_objective_function_runoff, 296
objective_q_et_tws_kge_catchment_avg
mo_objective_function, 400
objective_sm_corr
mo_objective_function, 402
objective_sm_kge_catchment_avg
mo_objective_function, 403
objective_sm_pd
mo_objective_function, 405
objective_sm_sse_standard_score
mo_objective_function, 406
objective_sse
mo_mrm_objective_function_runoff, 297
objective_sse_boxcox
mo_mrm_objective_function_runoff, 298
objective_subprocess
mo_objective_function, 407
objective_weighted_nse
mo_mrm_objective_function_runoff, 300
of
mo_mrm_write_fluxes_states::outputvariable, 512
mo_write_fluxes_states::outputvariable, 517
oldintegration
mo_neutrons, 384
opti_function
mo_common_mhm_mrm_variables, 104
opti_method
mo_common_mhm_mrm_variables, 104
optidata
mo_common_variables::domain_meta, 492
optimization
mo_optimization, 409
optimize
mo_common_mhm_mrm_variables, 104
optimize_restart
mo_common_mhm_mrm_variables, 104
output_deflate_level
mo_global_variables, 140
output_deflate_level_mrm
mo_mrm_global_variables, 249
output_double_precision
mo_global_variables, 140
output_double_precision_mrm
mo_mrm_global_variables, 249
outputflxstate
mo_global_variables, 140
outputflxstate_mrm
mo_mrm_global_variables, 249
overallnumberofdomains
mo_common_variables::domain_meta, 493

p1_initstatefluxes
mo_common_constants, 88
p2_initstatefluxes
mo_mhm_constants, 166
mo_mpr_constants, 177
p3_initstatefluxes
mo_mhm_constants, 167
mo_mpr_constants, 177
p4_initstatefluxes
mo_mhm_constants, 167
mo_mpr_constants, 177
p5_initstatefluxes
mo_mhm_constants, 167
mo_mpr_constants, 177
parameter_regularization
mo_mrm_objective_function_runoff, 301
peakdistribution
mo_mrm_signatures, 349
period_copy_period_data
mo_common_mhm_mrm_read_config, 98
pet_correctbyasp
mo_mpr_pet, 202
pet_correctbylai
mo_mpr_pet, 203
pet_hargreaves
mo_pet, 412
pet_penman
mo_pet, 413
pet_priestly
mo_pet, 415
pi_dp
mo_template, 462
pi_sp
mo_template, 462
prepare_gridded_daily_lai_data
mo_prepare_gridded_lai, 418
prepare_gridded_mean_monthly_lai_data
mo_prepare_gridded_lai, 419
prepare_meteo_forcings_data
mo_meteo_forcings, 156
prev_day
mo_common_datetime_type::datetimeinfo, 490
prev_month
mo_common_datetime_type::datetimeinfo, 490
prev_year
mo_common_datetime_type::datetimeinfo, 490

priestley_taylor_alpha
mo_mpr_pet, 204
print_startup_message
mo_mrm_init, 256
processmatrix
mo_common_variables, 123
project_details
mo_common_variables, 123
pt_a_water
mo_mrm_riv_temp_class::riv_temp_type, 536
pwp
mo_mpr_soilmoist, 222
pwp_c
mo_mpr_constants, 177
pwp_matpot_thetar
mo_mpr_constants, 177

q
mo_mrm_global_variables::gaugingstation, 496

read_const_nc
mo_read_nc, 425
read_data
mo_read_wrapper, 437
read_dem
mo_common_read_data, 112
read_geoformation_lut
mo_read_lut, 422
read_grid_info
mo_common_restart, 114
read_header_ascii
mo_read_spatial_data, 431
read_lai_lut
mo_read_lut, 423
read_latlon
mo_read_latlon, 420
read_lcover
mo_common_read_data, 113
read_meteo_weights
mo_global_variables, 140
read_mrm_routing_params
mo_mrm_read_config, 313
read_nc
mo_read_nc, 426
read_restart
mo_common_mhm_mrm_variables, 104
read_restart_states
mo_restart, 439
read_soil_lut
mo_soil_database, 449
read_spatial_data_ascii_dp
mo_read_spatial_data, 432
mo_read_spatial_data::read_spatial_data_ascii, 521
read_spatial_data_ascii_i4
mo_read_spatial_data, 433
mo_read_spatial_data::read_spatial_data_ascii, 522
read_timeseries
mo_read_timeseries, 434
read_weights_nc
mo_read_nc, 428
readoptidataobs
mo_read_optional_data, 430
readper
mo_common_mhm_mrm_variables, 105
reconstruct
mo_mrm_write_fluxes_states::outputvariable, 512
mo_write_fluxes_states::outputvariable, 517
reg_rout
mo_mrm_mpr, 260
reset_sum
mo_mrm_river_head, 337
reset_timestep
mo_mrm_riv_temp_class, 333
mo_mrm_riv_temp_class::riv_temp_type, 531
resolutionhydrology
mo_common_variables, 124
resolutionrouting
mo_common_mhm_mrm_variables, 105
right_bound
mo_common_variables::gridremapper, 500
riv_temp_pcs
mo_mrm_global_variables, 249
riv_widths_file
mo_mrm_riv_temp_class::riv_temp_type, 536
riv_widths_name
mo_mrm_riv_temp_class::riv_temp_type, 536
river_temp
mo_mrm_riv_temp_class::riv_temp_type, 536
rotate_fdir_variable
mo_mrm_read_data, 318
rout_space_weight
mo_mrm_constants, 229
routingstates
mo_global_variables, 141
runoff_sat_zone
mo_runoff, 444
runoff_unsat_zone
mo_runoff, 445
runoffratio
mo_mrm_signatures, 350
rzdepth
mo_mpr_global_variables::soiltype, 539

s11
mo_mrm_riv_temp_class::riv_temp_type, 537
sa_temp
mo_common_mhm_mrm_variables, 105
sand
mo_mpr_global_variables::soiltype, 540
sat_vap_pressure
mo_pet, 416
satpressureslope1
mo_mhm_constants, 167
sce_ngs
mo_common_mhm_mrm_variables, 105
sce_npg

mo_common_mhm_mrm_variables, 105
sce_nps
mo_common_mhm_mrm_variables, 106
seed
mo_common_mhm_mrm_variables, 106
set_domain_indices
mo_grid, 147
set_land_cover_scenes_id
mo_common_read_config, 110
setup_description
mo_common_variables, 124
simper
mo_common_mhm_mrm_variables, 106
simulation_type
mo_common_variables, 124
single_objective_runoff
mo_mrm_objective_function_runoff, 302
single_objective_runoff_master
mo_mrm_objective_function_runoff, 303
single_objective_runoff_subprocess
mo_mrm_objective_function_runoff, 304
slope_satpressure
mo_pet, 416
snow_acc_melt_param
mo_multi_param_reg, 376
snow_accum_melt
mo_snow_accum_melt, 447
soil_moisture
mo_soil_moisture, 452
soildb
mo_mpr_global_variables, 199
soilhorizonsvarname
mo_common_constants, 88
spatial_aggregation_3d
mo_spatial_agg_disagg_forcing, 455
mo_spatial_agg_disagg_forcing::spatial_aggregation, 541
spatial_aggregation_4d
mo_spatial_agg_disagg_forcing, 455
mo_spatial_agg_disagg_forcing::spatial_aggregation, 542
spatial_disaggregation_3d
mo_spatial_agg_disagg_forcing, 456
mo_spatial_agg_disagg_forcing::spatial_disaggregation, 543
spatial_disaggregation_4d
mo_spatial_agg_disagg_forcing, 456
mo_spatial_agg_disagg_forcing::spatial_disaggregation, 543
src/common/mo_check.f90, 549
src/common/mo_common_constants.f90, 550
src/common/mo_common_datetime_type.f90, 550
src/common/mo_common_file.f90, 551
src/common/mo_common_functions.f90, 551
src/common/mo_common_read_config.f90, 551
src/common/mo_common_read_data.f90, 552
src/common/mo_common_restart.f90, 552
src/common/mo_common_variables.f90, 552
src/common/mo_grid.f90, 553
src/common/mo_read_latlon.f90, 554
src/common/mo_read_nc.f90, 554
src/common/mo_read_spatial_data.f90, 555
src/common/mo_read_timeseries.f90, 555
src/common/mo_template.f90, 555
src/common_mHM_mRM/mo_common_mHM_mRM_file.f90, 556
src/common_mHM_mRM/mo_common_mHM_mRM_MPI_tools.f90, 556
src/common_mHM_mRM/mo_common_mHM_mRM_read_config.f90, 557
src/common_mHM_mRM/mo_common_mHM_mRM_restart.f90, 557
src/common_mHM_mRM/mo_common_mHM_mRM_variables.f90, 557
src/common_mHM_mRM/mo_optimization.f90, 558
src/mHM/mhm_driver.f90, 558
src/mHM/mo_canopy_interc.f90, 561
src/mHM/mo_file.f90, 562
src/mHM/mo_global_variables.f90, 562
src/mHM/mo_init_states.f90, 564
src/mHM/mo_meteo_forcings.f90, 564
src/mHM/mo_mhm.f90, 564
src/mHM/mo_mhm_constants.f90, 565
src/mHM/mo_mhm_eval.f90, 566
src/mHM/mo_mhm_read_config.f90, 566
src/mHM/mo_neutrons.f90, 566
src/mHM/mo_objective_function.f90, 567
src/mHM/mo_pet.f90, 567
src/mHM/mo_read_optional_data.f90, 568
src/mHM/mo_restart.f90, 568
src/mHM/mo_runoff.f90, 569
src/mHM/mo_snow_accum_melt.f90, 569
src/mHM/mo_soil_moisture.f90, 569
src/mHM/mo_spatial_agg_disagg_forcing.f90, 570
src/mHM/mo_startup.f90, 570
src/mHM/mo_temporal_disagg_forcing.f90, 570
src/mHM/mo_write_ascii.f90, 571
src/mHM/mo_write_fluxes_states.f90, 571
src/MPR/mo_mpr_constants.f90, 572
src/MPR/mo_mpr_eval.f90, 573
src/MPR/mo_mpr_file.f90, 573
src/MPR/mo_mpr_global_variables.f90, 574
src/MPR/mo_mpr_pet.f90, 576
src/MPR/mo_mpr_read_config.f90, 576
src/MPR/mo_mpr_restart.f90, 576
src/MPR/mo_mpr_runoff.f90, 577
src/MPR/mo_mpr_smhorizons.f90, 577
src/MPR/mo_mpr_soilmoist.f90, 577
src/MPR/mo_mpr_startup.f90, 578
src/MPR/mo_multi_param_reg.f90, 578
src/MPR/mo_prepare_gridded_lai.f90, 579
src/MPR/mo_read_lut.f90, 579
src/MPR/mo_read_wrapper.f90, 579
src/MPR/mo_soil_database.f90, 579
src/MPR/mo_upscaling_operators.f90, 580
src/MPR/mpr_driver.f90, 580

src/mRM/mo_mrm_constants.f90, 583
src/mRM/mo_mrm_file.f90, 583
src/mRM/mo_mrm_global_variables.f90, 584
src/mRM/mo_mrm_init.f90, 586
src/mRM/mo_mrm_mpr.f90, 586
src/mRM/mo_mrm_net_startup.f90, 586
src/mRM/mo_mrm_objective_function_runoff.f90, 588
src/mRM/mo_mrm_pre_routing.f90, 589
src/mRM/mo_mrm_read_config.f90, 589
src/mRM/mo_mrm_read_data.f90, 590
src/mRM/mo_mrm_restart.f90, 590
src/mRM/mo_mrm_riv_temp_class.f90, 590
src/mRM/mo_mrm_river_head.f90, 591
src/mRM/mo_mrm_routing.f90, 592
src/mRM/mo_mrm_signatures.f90, 592
src/mRM/mo_mrm_write.f90, 593
src/mRM/mo_mrm_write_fluxes_states.f90, 593
step_iter
    mo_mrm_riv_temp_class::riv_temp_type, 537
steps
    mo_mrm_write_fluxes_states::outputdataset, 504
    mo_write_fluxes_states::outputdataset, 508
store
    mo_mrm_write_fluxes_states::outputdataset, 504
    mo_mrm_write_fluxes_states::outputvariable, 512
    mo_write_fluxes_states::outputdataset, 508
    mo_write_fluxes_states::outputvariable, 517
sum_counter
    mo_mrm_river_head, 338

t
    mo_mrm_global_variables::gaugingstation, 496
tabularintegralafast
    mo_neutrons, 385
temporal_disagg_flux_daynight
    mo_temporal_disagg_forcing, 463
temporal_disagg_forcing
    mo_temporal_disagg_forcing, 464
temporal_disagg_meteo_weights
    mo_temporal_disagg_forcing, 466
temporal_disagg_state_daynight
    mo_temporal_disagg_forcing, 466
tetens_c1
    mo_mhm_constants, 167
tetens_c2
    mo_mhm_constants, 167
tetens_c3
    mo_mhm_constants, 168
the
    mo_mrm_write_fluxes_states::outputvariable, 512, 513
    mo_write_fluxes_states::outputvariable, 517
thetafc
    mo_mpr_global_variables::soiltype, 540
thetafc_till
    mo_mpr_global_variables::soiltype, 540
thetapw
    mo_mpr_global_variables::soiltype, 540
thetapw_till
    mo_mpr_global_variables::soiltype, 540
thetas
    mo_mpr_global_variables::soiltype, 540
thetas_till
    mo_mpr_global_variables::soiltype, 540
tillagedepth
    mo_mpr_global_variables, 200
time
    mo_mrm_write_fluxes_states::outputdataset, 504
    mo_write_fluxes_states::outputdataset, 508
time_counter
    mo_mrm_river_head, 338
timestep
    mo_common_mhm_mrm_variables, 106
timestep_lai_input
    mo_mpr_global_variables, 200
timestep_model_inputs
    mo_global_variables, 141
timestep_model_outputs
    mo_global_variables, 141
timestep_model_outputs_mrm
    mo_mrm_global_variables, 250
tindex_out
    mo_common_datetime_type::datetimeinfo, 491
to
    mo_mrm_write_fluxes_states::outputdataset, 505
    mo_mrm_write_fluxes_states::outputvariable, 513
    mo_write_fluxes_states::outputdataset, 508
    mo_write_fluxes_states::outputvariable, 518
ts_cnt
    mo_mrm_riv_temp_class::riv_temp_type, 537
turb_heat_ex_coeff
    mo_mrm_riv_temp_class::riv_temp_type, 537

uaspect
    mo_mpr_file, 187
uconfig
    mo_common_file, 91
    mo_mrm_file, 233
ud
    mo_mpr_global_variables::soiltype, 540
udaily_discharge
    mo_mrm_file, 233
udefoutput
    mo_file, 126
    mo_mrm_file, 233
udem
    mo_common_file, 91
udischarge
    mo_mrm_file, 233
ufacc
    mo_mrm_file, 234
ufdir
    mo_mrm_file, 234
ugaugeloc
    mo_mrm_file, 234
ugeolut
    mo_mpr_file, 187
uhydrogeoclass

mo_mpr_file, 187
ulaiclass
mo_mpr_file, 187
ulailut
mo_mpr_file, 187
ulcoverclass
mo_common_file, 91
umeteo
mo_mpr_file, 188
umeteo_header
mo_mpr_file, 188
unamelist_mhm
mo_file, 126
unamelist_mhm_param
mo_file, 126
unamelist_mpr
mo_mpr_file, 188
unamelist_mpr_param
mo_mpr_file, 188
unamelist_mrm
mo_mrm_file, 234
unamelist_param_mrm
mo_mrm_file, 234
unpack_field_and_write_1d_dp
mo_mpr_restart, 208
mo_mpr_restart::unpack_field_and_write, 544
mo_restart, 440
mo_restart::unpack_field_and_write, 546
unpack_field_and_write_1d_i4
mo_mpr_restart, 208
mo_mpr_restart::unpack_field_and_write, 544
mo_restart, 441
mo_restart::unpack_field_and_write, 546
unpack_field_and_write_2d_dp
mo_mpr_restart, 209
mo_mpr_restart::unpack_field_and_write, 545
mo_restart, 441
mo_restart::unpack_field_and_write, 546
unpack_field_and_write_3d_dp
mo_mpr_restart, 209
mo_mpr_restart::unpack_field_and_write, 545
mo_restart, 441
mo_restart::unpack_field_and_write, 547
uopti
mo_common_mhm_mrm_file, 93
uopti_nml
mo_common_mhm_mrm_file, 93
up_bnd_iter
mo_mrm_riv_temp_class::riv_temp_type, 537
up_iter
mo_mrm_riv_temp_class::riv_temp_type, 537
update_lai_timestep
mo_common_datetime_type::datetimeinfo, 489
updatedataset
mo_mrm_write_fluxes_states, 362
mo_mrm_write_fluxes_states::outputdataset, 503
mo_write_fluxes_states, 482
mo_write_fluxes_states::outputdataset, 506
updatevariable
mo_mrm_write_fluxes_states, 363
mo_mrm_write_fluxes_states::outputvariable, 510, 513
mo_write_fluxes_states, 483
mo_write_fluxes_states::outputvariable, 515, 518
upper_bound
mo_common_variables::gridremapper, 500
upscale_arithmetic_mean
mo_upscaling_operators, 470
upscale_geometric_mean
mo_upscaling_operators, 471
upscale_harmonic_mean
mo_upscaling_operators, 472
upscale_p_norm
mo_upscaling_operators, 473
uslope
mo_mpr_file, 188
mo_mrm_file, 234
usoil_database
mo_mpr_file, 189
usoilclass
mo_mpr_file, 189

variable
mo_mrm_write_fluxes_states::outputvariable, 513
mo_write_fluxes_states::outputvariable, 518
variables
mo_mrm_write_fluxes_states::outputdataset, 505
mo_write_fluxes_states::outputdataset, 508
variables_alloc
mo_init_states, 148
variables_alloc_routing
mo_mrm_init, 256
variables_default_init
mo_init_states, 149
variables_default_init_routing
mo_mrm_init, 257
varnametotalrunoff
mo_mrm_global_variables, 250
vars
mo_mrm_write_fluxes_states::outputdataset, 505
mo_write_fluxes_states::outputdataset, 509
version
mo_file, 126
mo_mpr_file, 189
mo_mrm_file, 235
version_date
mo_file, 126
mo_mpr_file, 189
mo_mrm_file, 235
vgenuchten_sandtresh
mo_mpr_constants, 177
vgenuchtenn_c1
mo_mpr_constants, 178
vgenuchtenn_c10
mo_mpr_constants, 178
vgenuchtenn_c11
mo_mpr_constants, 178

vgenuchtenn_c12
mo_mpr_constants, 178
vgenuchtenn_c13
mo_mpr_constants, 178
vgenuchtenn_c14
mo_mpr_constants, 178
vgenuchtenn_c15
mo_mpr_constants, 179
vgenuchtenn_c16
mo_mpr_constants, 179
vgenuchtenn_c17
mo_mpr_constants, 179
vgenuchtenn_c18
mo_mpr_constants, 179
vgenuchtenn_c2
mo_mpr_constants, 179
vgenuchtenn_c3
mo_mpr_constants, 179
vgenuchtenn_c4
mo_mpr_constants, 180
vgenuchtenn_c5
mo_mpr_constants, 180
vgenuchtenn_c6
mo_mpr_constants, 180
vgenuchtenn_c7
mo_mpr_constants, 180
vgenuchtenn_c8
mo_mpr_constants, 180
vgenuchtenn_c9
mo_mpr_constants, 180

warmingdays
mo_common_mhm_mrm_variables, 106
warmper
mo_common_mhm_mrm_variables, 107
wd
mo_mpr_global_variables::soiltype, 541
which
mo_mrm_write_fluxes_states::outputvariable, 513
mo_write_fluxes_states::outputvariable, 518
windmeasheight
mo_mpr_constants, 180
write
mo_mrm_write_fluxes_states::outputdataset, 505
mo_write_fluxes_states::outputdataset, 509
write_configfile
mo_mrm_write, 356
mo_write_ascii, 474
write_daily_obs_sim_discharge
mo_mrm_write, 357
write_eff_params
mo_mpr_restart, 209
write_grid_info
mo_common_restart, 115
write_mpr_restart_files
mo_mpr_restart, 210
write_optifile
mo_write_ascii, 475
write_optinamelist
mo_write_ascii, 476
write_restart
mo_common_variables, 124
write_restart_files
mo_restart, 441
writeout
mo_common_datetime_type::datetimeinfo, 489
writes
mo_mrm_write_fluxes_states::outputvariable, 513
mo_write_fluxes_states::outputvariable, 518
writetimestep
mo_mrm_write_fluxes_states, 364
mo_mrm_write_fluxes_states::outputdataset, 503
mo_write_fluxes_states, 484
mo_write_fluxes_states::outputdataset, 506
writevariableattributes
mo_mrm_write_fluxes_states, 364
mo_write_fluxes_states, 485
writevariabletimestep
mo_mrm_write_fluxes_states, 365
mo_mrm_write_fluxes_states::outputvariable, 510
mo_write_fluxes_states, 486
mo_write_fluxes_states::outputvariable, 515
writing
mo_mrm_write_fluxes_states::outputvariable, 514
mo_write_fluxes_states::outputvariable, 518
written
mo_mrm_write_fluxes_states::outputdataset, 505
mo_write_fluxes_states::outputdataset, 509

x
mo_common_variables::grid, 498
xllcorner
mo_common_variables::grid, 498

y
mo_common_variables::grid, 498
year
mo_common_datetime_type::datetimeinfo, 491
yend
mo_common_variables::period, 520
yid
mo_common_datetime_type::datetimeinfo, 491
yllcorner
mo_common_variables::grid, 498
ystart
mo_common_variables::period, 520

zeroflowratio
mo_mrm_signatures, 351