HyBayesCal

config_logging
SCRIPT_DIR: str
info_formatter: logging.Formatter
warn_formatter: logging.Formatter
error_formatter: logging.Formatter
logger: logging.getLogger("stochastic_calibration")
logger_warn: logging.getLogger("warnings")
logger_error: logging.getLogger("errors")
console_handler: logging.StreamHandler()
console_ehandler: logging.StreamHandler()
console_whandler: logging.StreamHandler()
info_handler=logging.FileHandler("logfile.log", "w")
warn_handler=logging.FileHandler("warnings.log", "w")
err_handler=logging.FileHandler("errors.log", "w")
import
Bal
observations: np.array
error: np.array
compute_likelihood(prediction:
np.array, normalize=False): np.array
compute_bayesian_scores(prediction:
np.array, method="weighting"): (float, float)
selection_criteria(al_strategy: str, al_BME:
np.array[d_size_AL], al_RE: np.array[d_size_AL]):
(float, int)
import
BalWithGPE
__numerical_model = None
__setattr__("numerical_model", software_coupling)
observations = {}
n_simulation = int
prior_distribution = np.array(())
collocation_points = np.ndarray(())
bme_csv_prior = ""
re_csv_prior = ""
bme_score_file = None
re_score_file = None
initialize_score_writing(func: function): function
full_model_calibration(): None
initiate_prior_distributions(): None
load_observations(): None
run_initial_simulations(): int
get_collocation_points(): None
get_surrogate_prediction(model_results:
np.array, number_of_points: int, prior:
np.array): (np.array(mean), np.array(std))
runBal(model_results: np.array, prior=np.array): int
sample_collocation_points(method="uniform"): None
import
function_pool
append_new_line(file_name: str,
text_to_append: str): None
call_subroutine(bash_command: str): int
calculate_settling_velocity(diameters:
np. array): np.array
concatenate_csv_pts(file_directory:
os.path, *args: str/list): pd.df
lookahead(iterable): bool
str2seq(list_like_string: str, separator=",",
return_type="tuple"): tuple/list
log_actions(func: function): function
import
import
import
model_structure
config_physics
GRAVITY: 9.81
KINEMATIC_VISCOSITY: 10 ** -6
WATER_DENSITY: 10. ** 3
SED_DENSITY: 2650
UserDefs
CALIB_PAR_SET: dict
CALIB_PTS: numpy
CALIB_TARGETS: list
init_runs: int
init_run_sampling: str
IT_LIMIT: int
MC_SAMPLES: int
MC_SAMPLES_AL: int
AL_SAMPLES: int
AL_STRATEGY: str
score_method: str
SIM_DIR: str
BME: None
RE: None
al_BME: None
al_RE: None
assign_global_settings(all attributes): None
check_user_input(None): None
read_wb_range(read_range: str, sheet_name: str):
pd.df
FullComplexityModel
model_dir: os.path
control_file: str
collocation_file: str
res_dir: str
update_model_controls(new_parameter_values:
dict, simulation_id: int): None
run_simulation(None): None
import
telemac
config_telemac
TM_TEMPLATE_DIR: os.path
GAIA_PARAMETERS: pd.df
TM2D_PARAMETERS: pd.df
TM_TRANSLATOR: dict
AL_RANGE = "A14:B22"
MEASUREMENT_DATA_RANGE = "A23:B26"
PRIOR_SCA_RANGE = "A32:B35"
PRIOR_VEC_RANGE = "A38:B40"
PRIOR_REC_RANGE = "A43:B44"
ZONAL_PAR_RANGE = "A47:A49"
RECALC_PARS: dict
import
TelemacUserDefs
N_CPUS: int
TM_CAS: str
tm_xD: str
GAIA_CAS: str
assign_calib_ranges(direct_par_df: pd.df,
vector_par_df: pd.df, recalc_par_df: pd.df): None
check_user_input(None): None
>> assign_global_settings
TelemacModel
calibration_parameters: dict
comm: mpi4py.MPI.Comm(mpi4py.MPI.COMM_WORLD)
control_file: str
nproc: int
slf_input_file: str
shebang: str
stdout: int (default: 6)
tm_cas: str
tm_results_file: str
tm_xd: str
tm_case: Telemac2() / Telemac3d()
tm_case_loaded: bool
tm_xd_dict: dict
**gaia_cas: str
**gaia_results_file: str
call_tm_shell(cmd: str): (subprocess.PIPE,
subprocess.Popen().returncode)
cmd2string(keyword: str): str.encode()
create_cas_string(param_name: str, value: var.): None
create_launcher_pyscript(filename: str): None
get_variable_value(slf_file_name: str,
calibration_par: str, specific_nodes:
np.array, save_name: str): np.array
load_case(): int
mpirun(filename: str): None
rename_selafin(old_name=".slf", new_name=".slf"): None
rewrite_steering_file(param_name: str,
updated_string: str, steering_module="telemac"): int
>> run_simulation(filename: str)
>> update_model_controls
import
pputils/ppmodules/selafin_io_pp
ppSELAFIN
float_type = 'f'
float_size = 4
title: str
precision = 'SELAFIN '
NBV1 = 0
NBV2 = 0
vars: list
vnames: list
vunits: list
IPARAM: list
NPLAN = 0
DATE = [1997, 8, 29, 2, 15, 0]
NELEM = 0
NPOIN = 0
NDP = 0
IKLE = np.zeros((self.NELEM,
self.NPOIN), dtype=np.int32)
IPOBO = np.zeros(self.NPOIN, dtype=np.int32)
x = np.zeros(self.NPOIN)
y = np.zeros(self.NPOIN)
time: list
temp = np.zeros((self.NBV1, self.NPOIN))
tempAtNode = np.zeros((0, 0))
readHeader(): None
writeHeader(): None
writeVariables(time: list,
temp: np.array((NBV1, NPOIN))
readTimes(): None
readVariables(t_des: int): None
readVariablesAtNode(node): None
setPrecision(ftype: str, fsize: int): None
getPrecision: (float_type, float_size), getNPOIN:
NPOIN, getNELEM: NELEM, getTimes: time, getVarNames:
vnames, getVarUnits: vunits, getNPLAN: int
NPLAN, getIKLE: IKLE, getMeshX: x, getMeshY:
y, getVarValues: temp, getVarValuesAtNode:
tempAtNode, getIPOBO: IPOBO, getDATE: DATE,
getMesh: (NELEM, NPOIN, NDP, IKLE, IPOBO, x, y)
setTitle(title: str), setDATE(DATE:list[6e]),
setVarNames(vnames:), setVarUnits(vunits:),
setIPARAM(IPARAM:),setMesh(NELEM: int, NPOIN: int,
NDP:int, IKLE: np.array, IPOBO: np.array, x: np.array,
y: np.array)
import