The mesoscale Hydrologic Model

mHM

Documentation for version 5.8

Edited by Luis Samaniego

in cooperation with

Johannes Brenner, Matthias Cuntz, Cüneyd M. Demirel, Maren Kaluza, Rohini Kumar, Ben Langenberg, Juliane Mai, Oldrich Rakovec, David Schäfer, Martin Schrön, Simon Stisen, Stephan Thober, Matthias Zink

HELMHOLTZ CENTRE FOR ENVIRONMENTAL RESEARCH – UFZ

Former mHM project collaborators

John Craven, Giovanni Dalmasso, Jude Musuuza, Vladyslav Prykhodko, Christoph Schneider, Diana Spieler

The mHM project

www.ufz.de/mhm

©2005-2017
Helmholtz Centre for Environmental Research - UFZ

Contents

1 Introduction to mHM **1**

1.1 Short Description . . . 1

1.2 The Grid-based mHM Model . . . 1

1.3 Model Formulation . . . 3

1.4 The Multiscale Parameter Regionalization Technique . . . 4

1.5 The Parameter Estimation Problem . . . 5

1.6 Model Calibration . . . 6

1.7 Helpful links . . . 7

1.8 Test basin . . . 7

1.9 Protocols . . . 7

2 Getting Started **9**

2.1 Receive mHM . . . 9

2.2 Install NETCDF . . . 9

2.2.1 Short Guide Install to NETCDF on Linux (example Ubuntu 16.04 with gfortran v5.4.0) . . . 9

2.2.2 Short Guide Install to NETCDF on MacOS . . . 10

2.3 Run mHM under CYGWIN on Windows . . . 11

2.4 Compile mHM . . . 14

2.5 Test mHM on Example Basin . . . 15

2.6 Run your own Simulation . . . 15

2.6.1 Main Configuration: Paths, Periods, Switches . . . 15

2.6.2 Output Configuration: Time Steps, States, Fluxes . . . 16

2.6.3 Regionalised Parameters: Initial Values and Ranges . . . 16

2.7 Calibration and Optimization . . . 16

2.7.1 The Optimization Routines . . . 16

2.7.2 Calibration Settings for mHM . . . 17

2.7.3 Final Calibration Results . . . 17

3 Data Preparation for mHM **19**

3.1 Getting Started . . . 19

3.1.1 Meteorological variables . . . 19

3.1.2 Morphological variables . . . 20

3.1.3 Land Cover . . . 20
3.1.4 Gauging Station Information . . . 20
3.1.5 Optional Data . . . 21
3.2 Preparation of the Forcings . . . 21
3.2.1 Extract Data to the Size of a Catchment . . . 21
3.2.2 Extact Data to the Time Period of Interest . . . 21
3.2.3 Data Types . . . 21
3.2.4 Variable Names . . . 22
3.2.5 The Header File . . . 22
3.2.6 NODATA values . . . 22
3.3 Preparation of the LatLon Grid . . . 22
3.4 Preparation of the Morphological Data . . . 23
3.5 A possible GIS workflow . . . 23
3.5.1 General considerations . . . 23
3.5.2 Slope map . . . 26
3.5.3 Aspect map . . . 27
3.5.4 Fill DEM sinks . . . 27
3.5.5 Flow direction and flow accumulation . . . 27
3.5.5.1 Spatial Analyst . . . 27
3.5.5.2 Arc Hydro Tools . . . 28
3.5.6 Gauges map . . . 29
3.5.7 Watershed delineation . . . 31
3.5.8 Mask the datasets . . . 32
3.5.9 Write the ascii grids . . . 32
3.6 Land Cover Data . . . 32
3.7 Table Data . . . 33
3.7.1 The soil look-up table . . . 33
3.7.2 The hydrogeolgy look-up table . . . 33
3.7.3 The LAI look-up table . . . 34
3.7.4 The gauge files . . . 34
3.8 Post-GIS preparation . . . 35

4 Visualizing Model Output **37**

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

6 Coding and Documentation Style **41**

7 The details about the test basin **45**

8 Protocols for setting up a new mHM basin **47**

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

8.2 Protocol for generating a restart file . . . 48

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

9 mRM **49**

10 mHM Dependencies **51**

11 Publications using mHM **53**

12 mHM RELEASE NOTES **57**

13 Data Type Index **65**

13.1 Data Types List . . . 65

14 File Index **71**

14.1 File List . . . 71

15 Data Type Documentation **73**

15.1 mo_moment::absdev Interface Reference . . . 73

15.1.1 Detailed Description . . . 73

15.1.2 Member Function/Subroutine Documentation . . . 73

15.1.2.1 absdev_dp . . . 73

15.1.2.2 absdev_sp . . . 73

15.2 mo_anneal::anneal Interface Reference . . . 74

15.2.1 Detailed Description . . . 74

15.2.2 Member Function/Subroutine Documentation . . . 76

15.2.2.1 anneal_dp . . . 76

15.3 mo_append::append Interface Reference . . . 76

15.3.1 Detailed Description . . . 77

15.3.2 Member Function/Subroutine Documentation . . . 77

15.3.2.1 append_char_3d . . . 77

15.3.2.2 append_char_m_m . . . 78

15.3.2.3 append_char_v_s . . . 78

15.3.2.4 append_char_v_v . . . 78

15.3.2.5 append_dp_3d . . . 78

15.3.2.6 append_dp_m_m . . . 78

15.3.2.7 append_dp_v_s . . . 78

15.3.2.8 append_dp_v_v . . . 78

15.3.2.9 append_i4_m_m . . . 78
15.3.2.10 append_i4_v_s . . . 78
15.3.2.11 append_i4_v_v . . . 78
15.3.2.12 append_i8_3d . . . 79
15.3.2.13 append_i8_m_m . . . 79
15.3.2.14 append_i8_v_s . . . 79
15.3.2.15 append_i8_v_v . . . 79
15.3.2.16 append_lgt_3d . . . 79
15.3.2.17 append_lgt_m_m . . . 79
15.3.2.18 append_lgt_v_s . . . 79
15.3.2.19 append_lgt_v_v . . . 79
15.3.2.20 append_sp_3d . . . 79
15.3.2.21 append_sp_m_m . . . 79
15.3.2.22 append_sp_v_s . . . 80
15.3.2.23 append_sp_v_v . . . 80
15.4 mo_corr::arth Interface Reference . . . 80
15.4.1 Detailed Description . . . 80
15.4.2 Member Function/Subroutine Documentation . . . 80
15.4.2.1 arth_dp . . . 80
15.4.2.2 arth_i4 . . . 80
15.4.2.3 arth_sp . . . 81
15.5 mo_ncwrite::attribute Type Reference . . . 81
15.5.1 Detailed Description . . . 82
15.5.2 Member Data Documentation . . . 82
15.5.2.1 name . . . 82
15.5.2.2 nvalues . . . 82
15.5.2.3 values . . . 82
15.5.2.4 xtype . . . 82
15.6 mo_corr::autocoeffk Interface Reference . . . 82
15.6.1 Detailed Description . . . 82
15.6.2 Member Function/Subroutine Documentation . . . 82
15.6.2.1 autocoeffk_1d_dp . . . 82
15.6.2.2 autocoeffk_1d_sp . . . 83
15.6.2.3 autocoeffk_dp . . . 83
15.6.2.4 autocoeffk_sp . . . 83
15.7 mo_corr::autocorr Interface Reference . . . 83
15.7.1 Detailed Description . . . 84
15.7.2 Member Function/Subroutine Documentation . . . 84
15.7.2.1 autocorr_1d_dp . . . 84
15.7.2.2 autocorr_1d_sp . . . 84

15.7.2.3 autocorr_dp . . . 84
15.7.2.4 autocorr_sp . . . 84
15.8 mo_moment::average Interface Reference . . . 85
15.8.1 Detailed Description . . . 85
15.8.2 Member Function/Subroutine Documentation . . . 85
15.8.2.1 average_dp . . . 85
15.8.2.2 average_sp . . . 86
15.9 mo_mrm_global_variables::basininfo_mrm Type Reference . . . 86
15.9.1 Detailed Description . . . 87
15.9.2 Member Data Documentation . . . 87
15.9.2.1 gaugeidlist . . . 87
15.9.2.2 gaugeindexlist . . . 87
15.9.2.3 gaugenodelist . . . 87
15.9.2.4 inflowgaugeheadwater . . . 87
15.9.2.5 inflowgaugeidlist . . . 87
15.9.2.6 inflowgaugeindexlist . . . 87
15.9.2.7 inflowgaugenodelist . . . 88
15.9.2.8 l0_coloutlet . . . 88
15.9.2.9 l0_noutlet . . . 88
15.9.2.10 l0_rowoutlet . . . 88
15.9.2.11 ngauges . . . 88
15.9.2.12 ninflowgauges . . . 88
15.10 mo_errormeasures::bias Interface Reference . . . 88
15.10.1 Detailed Description . . . 88
15.10.2 Member Function/Subroutine Documentation . . . 88
15.10.2.1 bias_dp_1d . . . 88
15.10.2.2 bias_dp_2d . . . 89
15.10.2.3 bias_dp_3d . . . 89
15.10.2.4 bias_sp_1d . . . 90
15.10.2.5 bias_sp_2d . . . 90
15.10.2.6 bias_sp_3d . . . 90
15.11 mo_moment::central_moment Interface Reference . . . 91
15.11.1 Detailed Description . . . 91
15.11.2 Member Function/Subroutine Documentation . . . 91
15.11.2.1 central_moment_dp . . . 91
15.11.2.2 central_moment_sp . . . 91
15.12 mo_moment::central_moment_var Interface Reference . . . 92
15.12.1 Detailed Description . . . 92
15.12.2 Member Function/Subroutine Documentation . . . 92
15.12.2.1 central_moment_var_dp . . . 92

15.12.2.2 central_moment_var_sp . . . 93
15.13 mo_standard_score::classified_standard_score Interface Reference . . . 93
15.13.1 Detailed Description . . . 93
15.13.2 Member Function/Subroutine Documentation . . . 94
15.13.2.1 classified_standard_score_dp . . . 94
15.13.2.2 classified_standard_score_sp . . . 95
15.14 mo_corr::corr Interface Reference . . . 95
15.14.1 Detailed Description . . . 95
15.14.2 Member Function/Subroutine Documentation . . . 95
15.14.2.1 corr_dp . . . 95
15.14.2.2 corr_sp . . . 96
15.15 mo_moment::correlation Interface Reference . . . 96
15.15.1 Detailed Description . . . 96
15.15.2 Member Function/Subroutine Documentation . . . 96
15.15.2.1 correlation_dp . . . 96
15.15.2.2 correlation_sp . . . 97
15.16 mo_moment::covariance Interface Reference . . . 97
15.16.1 Detailed Description . . . 97
15.16.2 Member Function/Subroutine Documentation . . . 97
15.16.2.1 covariance_dp . . . 98
15.16.2.2 covariance_sp . . . 98
15.17 mo_corr::crosscoeffk Interface Reference . . . 98
15.17.1 Detailed Description . . . 98
15.17.2 Member Function/Subroutine Documentation . . . 99
15.17.2.1 crosscoeffk_dp . . . 99
15.17.2.2 crosscoeffk_sp . . . 99
15.18 mo_corr::crosscorr Interface Reference . . . 99
15.18.1 Detailed Description . . . 100
15.18.2 Member Function/Subroutine Documentation . . . 100
15.18.2.1 crosscorr_dp . . . 100
15.18.2.2 crosscorr_sp . . . 100
15.19 mo_orderpack::ctrper Interface Reference . . . 100
15.19.1 Detailed Description . . . 101
15.19.2 Member Function/Subroutine Documentation . . . 101
15.19.2.1 d_ctrper . . . 101
15.19.2.2 i_ctrper . . . 101
15.19.2.3 r_ctrper . . . 101
15.20 mo_temporal_aggregation::day2mon_average Interface Reference . . . 101
15.20.1 Detailed Description . . . 101
15.20.2 Member Function/Subroutine Documentation . . . 102

15.20.2.1 day2mon_average_dp . . . 102
15.21 mo_ncwrite::dims Type Reference . . . 102
15.21.1 Detailed Description . . . 103
15.21.2 Member Data Documentation . . . 103
15.21.2.1 dimid . . . 103
15.21.2.2 len . . . 103
15.21.2.3 name . . . 103
15.22 dummy_mrm Module Reference . . . 103
15.22.1 Detailed Description . . . 103
15.23 mo_ncwrite::dump_netcdf Interface Reference . . . 103
15.23.1 Detailed Description . . . 104
15.23.2 Member Function/Subroutine Documentation . . . 104
15.23.2.1 dump_netcdf_1d_dp . . . 104
15.23.2.2 dump_netcdf_1d_i4 . . . 104
15.23.2.3 dump_netcdf_1d_sp . . . 104
15.23.2.4 dump_netcdf_2d_dp . . . 104
15.23.2.5 dump_netcdf_2d_i4 . . . 104
15.23.2.6 dump_netcdf_2d_sp . . . 104
15.23.2.7 dump_netcdf_3d_dp . . . 105
15.23.2.8 dump_netcdf_3d_i4 . . . 105
15.23.2.9 dump_netcdf_3d_sp . . . 105
15.23.2.10 dump_netcdf_4d_dp . . . 105
15.23.2.11 dump_netcdf_4d_i4 . . . 105
15.23.2.12 dump_netcdf_4d_sp . . . 105
15.23.2.13 dump_netcdf_5d_dp . . . 105
15.23.2.14 dump_netcdf_5d_i4 . . . 105
15.23.2.15 dump_netcdf_5d_sp . . . 106
15.24 mo_utils::eq Interface Reference . . . 106
15.24.1 Detailed Description . . . 106
15.24.2 Member Function/Subroutine Documentation . . . 106
15.24.2.1 equal_dp . . . 106
15.24.2.2 equal_sp . . . 106
15.25 mo_utils::equal Interface Reference . . . 106
15.25.1 Detailed Description . . . 106
15.25.2 Member Function/Subroutine Documentation . . . 107
15.25.2.1 equal_dp . . . 107
15.25.2.2 equal_sp . . . 107
15.26 mo_optimization_utils::eval_interface Interface Reference . . . 108
15.26.1 Detailed Description . . . 108
15.26.2 Constructor & Destructor Documentation . . . 108

15.26.2.1 eval_interface . . . 108
15.27 mo_orderpack::fndnth Interface Reference . . . 108
15.27.1 Detailed Description . . . 108
15.27.2 Member Function/Subroutine Documentation . . . 108
15.27.2.1 d_fndnth . . . 108
15.27.2.2 i_fndnth . . . 108
15.27.2.3 r_fndnth . . . 109
15.28 mo_corr::four1 Interface Reference . . . 109
15.28.1 Detailed Description . . . 109
15.28.2 Member Function/Subroutine Documentation . . . 109
15.28.2.1 four1_dp . . . 109
15.28.2.2 four1_sp . . . 109
15.29 mo_corr::fourrow Interface Reference . . . 109
15.29.1 Detailed Description . . . 109
15.29.2 Member Function/Subroutine Documentation . . . 110
15.29.2.1 fourrow_dp . . . 110
15.29.2.2 fourrow_sp . . . 110
15.30 mo_mrm_global_variables::gaugingstation Type Reference . . . 110
15.30.1 Detailed Description . . . 110
15.30.2 Member Data Documentation . . . 111
15.30.2.1 basinid . . . 111
15.30.2.2 fname . . . 111
15.30.2.3 gaugeid . . . 111
15.30.2.4 q . . . 111
15.31 mo_utils::ge Interface Reference . . . 111
15.31.1 Detailed Description . . . 111
15.31.2 Member Function/Subroutine Documentation . . . 111
15.31.2.1 greaterequal_dp . . . 111
15.31.2.2 greaterequal_sp . . . 111
15.32 mo_anneal::generate_neighborhood_weight Interface Reference . . . 111
15.32.1 Detailed Description . . . 112
15.32.2 Member Function/Subroutine Documentation . . . 112
15.32.2.1 generate_neighborhood_weight_dp . . . 112
15.33 mo_ncread::get_ncvar Interface Reference . . . 112
15.33.1 Detailed Description . . . 113
15.33.2 Member Function/Subroutine Documentation . . . 113
15.33.2.1 get_ncvar_0d_dp . . . 113
15.33.2.2 get_ncvar_0d_i1 . . . 113
15.33.2.3 get_ncvar_0d_i4 . . . 113
15.33.2.4 get_ncvar_0d_sp . . . 113

15.33.2.5 get_ncvar_1d_dp . . . 113
15.33.2.6 get_ncvar_1d_i1 . . . 113
15.33.2.7 get_ncvar_1d_i4 . . . 114
15.33.2.8 get_ncvar_1d_sp . . . 114
15.33.2.9 get_ncvar_2d_dp . . . 114
15.33.2.10 get_ncvar_2d_i1 . . . 114
15.33.2.11 get_ncvar_2d_i4 . . . 114
15.33.2.12 get_ncvar_2d_sp . . . 114
15.33.2.13 get_ncvar_3d_dp . . . 114
15.33.2.14 get_ncvar_3d_i1 . . . 114
15.33.2.15 get_ncvar_3d_i4 . . . 115
15.33.2.16 get_ncvar_3d_sp . . . 115
15.33.2.17 get_ncvar_4d_dp . . . 115
15.33.2.18 get_ncvar_4d_i1 . . . 115
15.33.2.19 get_ncvar_4d_i4 . . . 115
15.33.2.20 get_ncvar_4d_sp . . . 115
15.33.2.21 get_ncvar_5d_dp . . . 115
15.33.2.22 get_ncvar_5d_i1 . . . 115
15.33.2.23 get_ncvar_5d_i4 . . . 116
15.33.2.24 get_ncvar_5d_sp . . . 116
15.34 mo_xor4096::get_timeseed Interface Reference . . . 116
15.34.1 Detailed Description . . . 116
15.34.2 Member Function/Subroutine Documentation . . . 116
15.34.2.1 get_timeseed_i4_0d . . . 116
15.34.2.2 get_timeseed_i4_1d . . . 116
15.34.2.3 get_timeseed_i8_0d . . . 116
15.34.2.4 get_timeseed_i8_1d . . . 116
15.35 mo_anneal::gettemperature Interface Reference . . . 117
15.35.1 Detailed Description . . . 117
15.35.2 Member Function/Subroutine Documentation . . . 118
15.35.2.1 gettemperature_dp . . . 118
15.36 mo_utils::greaterequal Interface Reference . . . 118
15.36.1 Detailed Description . . . 119
15.36.2 Member Function/Subroutine Documentation . . . 119
15.36.2.1 greaterequal_dp . . . 119
15.36.2.2 greaterequal_sp . . . 119
15.37 mo_common_variables::grid Type Reference . . . 120
15.37.1 Detailed Description . . . 120
15.37.2 Member Data Documentation . . . 121
15.37.2.1 cellarea . . . 121

15.37.2.2 cellcoor . . . 121
15.37.2.3 cellsize . . . 121
15.37.2.4 id . . . 121
15.37.2.5 iend . . . 121
15.37.2.6 istart . . . 121
15.37.2.7 mask . . . 121
15.37.2.8 ncells . . . 121
15.37.2.9 ncols . . . 121
15.37.2.10 nodata_value . . . 121
15.37.2.11 nrows . . . 121
15.37.2.12 x . . . 121
15.37.2.13 xllcorner . . . 122
15.37.2.14 y . . . 122
15.37.2.15 yllcorner . . . 122
15.38 mo_common_variables::gridremapper Type Reference . . . 122
15.38.1 Detailed Description . . . 123
15.38.2 Member Data Documentation . . . 123
15.38.2.1 high_res_grid . . . 123
15.38.2.2 left_bound . . . 123
15.38.2.3 low_res_grid . . . 123
15.38.2.4 lower_bound . . . 123
15.38.2.5 lowres_id_on_highres . . . 123
15.38.2.6 n_subcells . . . 123
15.38.2.7 right_bound . . . 123
15.38.2.8 upper_bound . . . 124
15.39 mo_temporal_aggregation::hour2day_average Interface Reference . . . 124
15.39.1 Detailed Description . . . 124
15.39.2 Member Function/Subroutine Documentation . . . 125
15.39.2.1 hour2day_average_dp . . . 125
15.40 mo_orderpack::indmed Interface Reference . . . 125
15.40.1 Detailed Description . . . 125
15.40.2 Member Function/Subroutine Documentation . . . 125
15.40.2.1 d_indmed . . . 125
15.40.2.2 i_indmed . . . 125
15.40.2.3 r_indmed . . . 125
15.41 mo_orderpack::indnth Interface Reference . . . 125
15.41.1 Detailed Description . . . 126
15.41.2 Member Function/Subroutine Documentation . . . 126
15.41.2.1 d_indnth . . . 126
15.41.2.2 i_indnth . . . 126

15.41.2.3 r_indnth . . . 126
15.42 mo_orderpack::inspar Interface Reference . . . 126
15.42.1 Detailed Description . . . 126
15.42.2 Member Function/Subroutine Documentation . . . 126
15.42.2.1 d_inspar . . . 126
15.42.2.2 i_inspar . . . 126
15.42.2.3 r_inspar . . . 127
15.43 mo_orderpack::inssor Interface Reference . . . 127
15.43.1 Detailed Description . . . 127
15.43.2 Member Function/Subroutine Documentation . . . 127
15.43.2.1 d_inssor . . . 127
15.43.2.2 i_inssor . . . 127
15.43.2.3 r_inssor . . . 128
15.44 mo_utils::is_finite Interface Reference . . . 128
15.44.1 Detailed Description . . . 128
15.44.2 Member Function/Subroutine Documentation . . . 129
15.44.2.1 is_finite_dp . . . 129
15.44.2.2 is_finite_sp . . . 129
15.45 mo_utils::is_nan Interface Reference . . . 130
15.45.1 Detailed Description . . . 130
15.45.2 Member Function/Subroutine Documentation . . . 130
15.45.2.1 is_nan_dp . . . 130
15.45.2.2 is_nan_sp . . . 130
15.46 mo_utils::is_normal Interface Reference . . . 131
15.46.1 Detailed Description . . . 131
15.46.2 Member Function/Subroutine Documentation . . . 131
15.46.2.1 is_normal_dp . . . 131
15.46.2.2 is_normal_sp . . . 131
15.47 mo_errormeasures::kge Interface Reference . . . 132
15.47.1 Detailed Description . . . 132
15.47.2 Member Function/Subroutine Documentation . . . 133
15.47.2.1 kge_dp_1d . . . 133
15.47.2.2 kge_dp_2d . . . 133
15.47.2.3 kge_dp_3d . . . 133
15.47.2.4 kge_sp_1d . . . 134
15.47.2.5 kge_sp_2d . . . 134
15.47.2.6 kge_sp_3d . . . 135
15.48 mo_errormeasures::kgenocorr Interface Reference . . . 135
15.48.1 Detailed Description . . . 135
15.48.2 Member Function/Subroutine Documentation . . . 136

15.48.2.1 kgenocorr_dp_1d . . . 136
15.48.2.2 kgenocorr_dp_2d . . . 136
15.48.2.3 kgenocorr_dp_3d . . . 137
15.48.2.4 kgenocorr_sp_1d . . . 137
15.48.2.5 kgenocorr_sp_2d . . . 137
15.48.2.6 kgenocorr_sp_3d . . . 138
15.49 mo_moment::kurtosis Interface Reference . . . 138
15.49.1 Detailed Description . . . 138
15.49.2 Member Function/Subroutine Documentation . . . 138
15.49.2.1 kurtosis_dp . . . 138
15.49.2.2 kurtosis_sp . . . 139
15.50 mo_utils::le Interface Reference . . . 139
15.50.1 Detailed Description . . . 139
15.50.2 Member Function/Subroutine Documentation . . . 139
15.50.2.1 lesserequal_dp . . . 140
15.50.2.2 lesserequal_sp . . . 140
15.51 mo_utils::lesserequal Interface Reference . . . 140
15.51.1 Detailed Description . . . 140
15.51.2 Member Function/Subroutine Documentation . . . 140
15.51.2.1 lesserequal_dp . . . 140
15.51.2.2 lesserequal_sp . . . 140
15.52 mo_linfit::linfit Interface Reference . . . 141
15.52.1 Detailed Description . . . 141
15.52.2 Member Function/Subroutine Documentation . . . 142
15.52.2.1 linfit_dp . . . 142
15.52.2.2 linfit_sp . . . 142
15.53 mo_errormeasures::lnnse Interface Reference . . . 143
15.53.1 Detailed Description . . . 143
15.53.2 Member Function/Subroutine Documentation . . . 143
15.53.2.1 lnnse_dp_1d . . . 143
15.53.2.2 lnnse_dp_2d . . . 144
15.53.2.3 lnnse_dp_3d . . . 144
15.53.2.4 lnnse_sp_1d . . . 144
15.53.2.5 lnnse_sp_2d . . . 145
15.53.2.6 lnnse_sp_3d . . . 145
15.54 mo_utils::locate Interface Reference . . . 145
15.54.1 Detailed Description . . . 146
15.54.2 Member Function/Subroutine Documentation . . . 146
15.54.2.1 locate_0d_dp . . . 146
15.54.2.2 locate_0d_sp . . . 147

15.54.2.3 locate_1d_dp . . . 147
15.54.2.4 locate_1d_sp . . . 147
15.55 mo_mad::mad Interface Reference . . . 148
15.55.1 Detailed Description . . . 148
15.55.2 Member Function/Subroutine Documentation . . . 148
15.55.2.1 mad_dp . . . 148
15.55.2.2 mad_sp . . . 149
15.55.2.3 mad_val_dp . . . 149
15.55.2.4 mad_val_sp . . . 149
15.56 mo_errormeasures::mae Interface Reference . . . 150
15.56.1 Detailed Description . . . 150
15.56.2 Member Function/Subroutine Documentation . . . 150
15.56.2.1 mae_dp_1d . . . 150
15.56.2.2 mae_dp_2d . . . 150
15.56.2.3 mae_dp_3d . . . 151
15.56.2.4 mae_sp_1d . . . 151
15.56.2.5 mae_sp_2d . . . 152
15.56.2.6 mae_sp_3d . . . 152
15.57 mo_mcmc::mcmc Interface Reference . . . 152
15.57.1 Detailed Description . . . 153
15.57.2 Member Function/Subroutine Documentation . . . 156
15.57.2.1 mcmc_dp . . . 156
15.58 mo_mcmc::mcmc_stddev Interface Reference . . . 156
15.58.1 Detailed Description . . . 157
15.58.2 Member Function/Subroutine Documentation . . . 160
15.58.2.1 mcmc_stddev_dp . . . 160
15.59 mo_template::mean Interface Reference . . . 160
15.59.1 Detailed Description . . . 160
15.59.2 Member Function/Subroutine Documentation . . . 161
15.59.2.1 mean_dp . . . 161
15.59.2.2 mean_sp . . . 161
15.60 mo_moment::mean Interface Reference . . . 162
15.60.1 Detailed Description . . . 162
15.60.2 Member Function/Subroutine Documentation . . . 162
15.60.2.1 mean_dp . . . 162
15.60.2.2 mean_sp . . . 162
15.61 mo_percentile::median Interface Reference . . . 162
15.61.1 Detailed Description . . . 163
15.61.2 Member Function/Subroutine Documentation . . . 163
15.61.2.1 median_dp . . . 163

15.61.2.2 median_sp . . . 163
15.62 mo_moment::mixed_central_moment Interface Reference . . . 163
15.62.1 Detailed Description . . . 164
15.62.2 Member Function/Subroutine Documentation . . . 164
15.62.2.1 mixed_central_moment_dp . . . 164
15.62.2.2 mixed_central_moment_sp . . . 164
15.63 mo_moment::mixed_central_moment_var Interface Reference . . . 164
15.63.1 Detailed Description . . . 165
15.63.2 Member Function/Subroutine Documentation . . . 165
15.63.2.1 mixed_central_moment_var_dp . . . 165
15.63.2.2 mixed_central_moment_var_sp . . . 165
15.64 mo_anneal Module Reference . . . 166
15.64.1 Detailed Description . . . 166
15.64.2 Member Function/Subroutine Documentation . . . 166
15.64.2.1 anneal_dp . . . 166
15.64.2.2 dchange_dp . . . 167
15.64.2.3 generate_neighborhood_weight_dp . . . 167
15.64.2.4 gettemperature_dp . . . 167
15.64.2.5 pargen_anneal_dp . . . 168
15.64.2.6 pargen_dds_dp . . . 168
15.65 mo_append Module Reference . . . 169
15.65.1 Detailed Description . . . 170
15.65.2 Member Function/Subroutine Documentation . . . 170
15.65.2.1 append_char_3d . . . 170
15.65.2.2 append_char_m_m . . . 170
15.65.2.3 append_char_v_s . . . 170
15.65.2.4 append_char_v_v . . . 171
15.65.2.5 append_dp_3d . . . 171
15.65.2.6 append_dp_m_m . . . 171
15.65.2.7 append_dp_v_s . . . 171
15.65.2.8 append_dp_v_v . . . 171
15.65.2.9 append_i4_3d . . . 171
15.65.2.10 append_i4_m_m . . . 171
15.65.2.11 append_i4_v_s . . . 171
15.65.2.12 append_i4_v_v . . . 171
15.65.2.13 append_i8_3d . . . 171
15.65.2.14 append_i8_m_m . . . 172
15.65.2.15 append_i8_v_s . . . 172
15.65.2.16 append_i8_v_v . . . 172
15.65.2.17 append_lgt_3d . . . 172

15.65.2.18 append_lgt_m_m . . . 172
15.65.2.19 append_lgt_v_s . . . 172
15.65.2.20 append_lgt_v_v . . . 172
15.65.2.21 append_sp_3d . . . 172
15.65.2.22 append_sp_m_m . . . 172
15.65.2.23 append_sp_v_s . . . 172
15.65.2.24 append_sp_v_v . . . 173
15.65.2.25 paste_char_m_m . . . 173
15.65.2.26 paste_char_m_s . . . 173
15.65.2.27 paste_char_m_v . . . 173
15.65.2.28 paste_dp_m_m . . . 173
15.65.2.29 paste_dp_m_s . . . 173
15.65.2.30 paste_dp_m_v . . . 173
15.65.2.31 paste_i4_m_m . . . 173
15.65.2.32 paste_i4_m_s . . . 173
15.65.2.33 paste_i4_m_v . . . 173
15.65.2.34 paste_i8_m_m . . . 174
15.65.2.35 paste_i8_m_s . . . 174
15.65.2.36 paste_i8_m_v . . . 174
15.65.2.37 paste_lgt_m_m . . . 174
15.65.2.38 paste_lgt_m_s . . . 174
15.65.2.39 paste_lgt_m_v . . . 174
15.65.2.40 paste_sp_m_m . . . 174
15.65.2.41 paste_sp_m_s . . . 174
15.65.2.42 paste_sp_m_v . . . 174
15.66 mo_canopy_interc Module Reference . . . 175
15.66.1 Detailed Description . . . 175
15.66.2 Member Function/Subroutine Documentation . . . 175
15.66.2.1 canopy_interc . . . 175
15.67 mo_common_constants Module Reference . . . 176
15.67.1 Detailed Description . . . 177
15.67.2 Member Data Documentation . . . 178
15.67.2.1 dayhours . . . 178
15.67.2.2 daysecs . . . 178
15.67.2.3 eps_dp . . . 178
15.67.2.4 eps_sp . . . 178
15.67.2.5 hoursecs . . . 178
15.67.2.6 maxnlcovers . . . 178
15.67.2.7 maxnobasins . . . 178
15.67.2.8 ncolpars . . . 178

15.67.2.9 nodata_dp . . . 178
15.67.2.10 nodata_i4 . . . 178
15.67.2.11 p1_initstatefluxes . . . 179
15.67.2.12 yeardays . . . 179
15.67.2.13 yearmonths . . . 179
15.67.2.14 yearmonths_i4 . . . 179
15.68 mo_common_file Module Reference . . . 179
15.68.1 Detailed Description . . . 180
15.68.2 Member Data Documentation . . . 180
15.68.2.1 file_config . . . 180
15.68.2.2 file_dem . . . 180
15.68.2.3 uconfig . . . 180
15.68.2.4 udem . . . 180
15.68.2.5 ulcoverclass . . . 180
15.69 mo_common_functions Module Reference . . . 181
15.69.1 Detailed Description . . . 181
15.69.2 Member Function/Subroutine Documentation . . . 181
15.69.2.1 in_bound . . . 181
15.70 mo_common_read_config Module Reference . . . 182
15.70.1 Detailed Description . . . 182
15.70.2 Member Function/Subroutine Documentation . . . 182
15.70.2.1 common_read_config . . . 182
15.70.2.2 set_land_cover_scenes_id . . . 183
15.71 mo_common_read_data Module Reference . . . 184
15.71.1 Detailed Description . . . 184
15.71.2 Member Function/Subroutine Documentation . . . 185
15.71.2.1 read_dem . . . 185
15.71.2.2 read_lcover . . . 185
15.72 mo_common_restart Module Reference . . . 186
15.72.1 Detailed Description . . . 187
15.72.2 Member Function/Subroutine Documentation . . . 187
15.72.2.1 read_grid_info . . . 187
15.72.2.2 write_grid_info . . . 188
15.73 mo_common_variables Module Reference . . . 189
15.73.1 Detailed Description . . . 191
15.73.2 Member Data Documentation . . . 191
15.73.2.1 alma_convention . . . 191
15.73.2.2 contact . . . 191
15.73.2.3 conventions . . . 191
15.73.2.4 dircommonfiles . . . 191

15.73.2.5 dirconfigout . . . 191
15.73.2.6 dirlcover . . . 191
15.73.2.7 dirmorpho . . . 192
15.73.2.8 dirout . . . 192
15.73.2.9 dirrestartout . . . 192
15.73.2.10 filelatlon . . . 192
15.73.2.11 global_parameters . . . 192
15.73.2.12 global_parameters_name . . . 192
15.73.2.13 history . . . 192
15.73.2.14 iflag_cordinate_sys . . . 192
15.73.2.15 l0_basin . . . 192
15.73.2.16 l0_elev . . . 192
15.73.2.17 l0_l11_remap . . . 192
15.73.2.18 l0_l1_remap . . . 192
15.73.2.19 l0_lcover . . . 193
15.73.2.20 l11_basin . . . 193
15.73.2.21 l1_l11_remap . . . 193
15.73.2.22 lc_year_end . . . 193
15.73.2.23 lc_year_start . . . 193
15.73.2.24 lcfilename . . . 193
15.73.2.25 level0 . . . 193
15.73.2.26 level1 . . . 193
15.73.2.27 mhm_details . . . 193
15.73.2.28 nbasins . . . 193
15.73.2.29 nlcoverscene . . . 193
15.73.2.30 nprocesses . . . 193
15.73.2.31 nuniquel0basins . . . 194
15.73.2.32 processmatrix . . . 194
15.73.2.33 project_details . . . 194
15.73.2.34 resolutionhydrology . . . 194
15.73.2.35 setup_description . . . 194
15.73.2.36 simulation_type . . . 194
15.73.2.37 write_restart . . . 194
15.74 mo_constants Module Reference . . . 194
15.74.1 Detailed Description . . . 197
15.74.2 Member Data Documentation . . . 197
15.74.2.1 cp0_dp . . . 197
15.74.2.2 cp0_sp . . . 197
15.74.2.3 dayhours . . . 198
15.74.2.4 daysecs . . . 198

15.74.2.5 deg2rad_dp . . . 198
15.74.2.6 deg2rad_sp . . . 198
15.74.2.7 euler . . . 198
15.74.2.8 euler_d . . . 198
15.74.2.9 gravity_dp . . . 198
15.74.2.10 gravity_sp . . . 198
15.74.2.11 nerr . . . 198
15.74.2.12 nin . . . 198
15.74.2.13 nnml . . . 199
15.74.2.14 nout . . . 199
15.74.2.15 p0_dp . . . 199
15.74.2.16 p0_sp . . . 199
15.74.2.17 pi . . . 199
15.74.2.18 pi_d . . . 199
15.74.2.19 pi_dp . . . 199
15.74.2.20 pi_sp . . . 199
15.74.2.21 pio2 . . . 199
15.74.2.22 pio2_d . . . 200
15.74.2.23 pio2_dp . . . 200
15.74.2.24 pio2_sp . . . 200
15.74.2.25 psychro_dp . . . 200
15.74.2.26 psychro_sp . . . 200
15.74.2.27 rad2deg_dp . . . 200
15.74.2.28 rad2deg_sp . . . 200
15.74.2.29 radiusearth_dp . . . 200
15.74.2.30 radiusearth_sp . . . 200
15.74.2.31 rho0_dp . . . 200
15.74.2.32 rho0_sp . . . 201
15.74.2.33 secday_dp . . . 201
15.74.2.34 secday_sp . . . 201
15.74.2.35 sigma_dp . . . 201
15.74.2.36 sigma_sp . . . 201
15.74.2.37 solarconst_dp . . . 201
15.74.2.38 solarconst_sp . . . 201
15.74.2.39 specheatet_dp . . . 201
15.74.2.40 specheatet_sp . . . 201
15.74.2.41 sqrt2 . . . 202
15.74.2.42 sqrt2_d . . . 202
15.74.2.43 sqrt2_dp . . . 202
15.74.2.44 sqrt2_sp . . . 202

15.74.2.45 t0_dp . . . 202
15.74.2.46 t0_sp . . . 202
15.74.2.47 twopi . . . 202
15.74.2.48 twopi_d . . . 202
15.74.2.49 twopi_dp . . . 202
15.74.2.50 twopi_sp . . . 203
15.74.2.51 twothird_dp . . . 203
15.74.2.52 twothird_sp . . . 203
15.74.2.53 yeardays . . . 203
15.74.2.54 yearmonths . . . 203
15.75 mo_corr Module Reference . . . 203
15.75.1 Detailed Description . . . 204
15.75.2 Member Function/Subroutine Documentation . . . 205
15.75.2.1 arth_dp . . . 205
15.75.2.2 arth_i4 . . . 205
15.75.2.3 arth_sp . . . 205
15.75.2.4 autocoeffk_1d_dp . . . 206
15.75.2.5 autocoeffk_1d_sp . . . 206
15.75.2.6 autocoeffk_dp . . . 206
15.75.2.7 autocoeffk_sp . . . 206
15.75.2.8 autocorr_1d_dp . . . 207
15.75.2.9 autocorr_1d_sp . . . 207
15.75.2.10 autocorr_dp . . . 207
15.75.2.11 autocorr_sp . . . 207
15.75.2.12 corr_dp . . . 208
15.75.2.13 corr_sp . . . 208
15.75.2.14 crosscoeffk_dp . . . 209
15.75.2.15 crosscoeffk_sp . . . 209
15.75.2.16 crosscorr_dp . . . 209
15.75.2.17 crosscorr_sp . . . 210
15.75.2.18 four1_dp . . . 210
15.75.2.19 four1_sp . . . 210
15.75.2.20 fourrow_dp . . . 210
15.75.2.21 fourrow_sp . . . 210
15.75.2.22 realft_dp . . . 211
15.75.2.23 realft_sp . . . 211
15.75.2.24 swap_1d_dpc . . . 211
15.75.2.25 swap_1d_spc . . . 211
15.75.2.26 zroots_unity_dp . . . 211
15.75.2.27 zroots_unity_sp . . . 212

15.75.3 Member Data Documentation . . . 212
15.75.3.1 npar2_arth . . . 212
15.75.3.2 npar_arth . . . 212
15.76 mo_dds Module Reference . . . 212
15.76.1 Detailed Description . . . 213
15.76.2 Member Function/Subroutine Documentation . . . 213
15.76.2.1 dds . . . 213
15.76.2.2 mdds . . . 214
15.76.2.3 neigh_value . . . 216
15.77 mo_errormeasures Module Reference . . . 216
15.77.1 Detailed Description . . . 218
15.77.2 Member Function/Subroutine Documentation . . . 218
15.77.2.1 bias_dp_1d . . . 218
15.77.2.2 bias_dp_2d . . . 219
15.77.2.3 bias_dp_3d . . . 219
15.77.2.4 bias_sp_1d . . . 220
15.77.2.5 bias_sp_2d . . . 220
15.77.2.6 bias_sp_3d . . . 220
15.77.2.7 kge_dp_1d . . . 221
15.77.2.8 kge_dp_2d . . . 221
15.77.2.9 kge_dp_3d . . . 222
15.77.2.10 kge_sp_1d . . . 222
15.77.2.11 kge_sp_2d . . . 222
15.77.2.12 kge_sp_3d . . . 223
15.77.2.13 kgenocorr_dp_1d . . . 223
15.77.2.14 kgenocorr_dp_2d . . . 224
15.77.2.15 kgenocorr_dp_3d . . . 224
15.77.2.16 kgenocorr_sp_1d . . . 224
15.77.2.17 kgenocorr_sp_2d . . . 225
15.77.2.18 kgenocorr_sp_3d . . . 225
15.77.2.19 lnnse_dp_1d . . . 225
15.77.2.20 lnnse_dp_2d . . . 226
15.77.2.21 lnnse_dp_3d . . . 226
15.77.2.22 lnnse_sp_1d . . . 226
15.77.2.23 lnnse_sp_2d . . . 227
15.77.2.24 lnnse_sp_3d . . . 227
15.77.2.25 mae_dp_1d . . . 227
15.77.2.26 mae_dp_2d . . . 228
15.77.2.27 mae_dp_3d . . . 228
15.77.2.28 mae_sp_1d . . . 229

15.77.2.29 mae_sp_2d . . . 229
15.77.2.30 mae_sp_3d . . . 230
15.77.2.31 mse_dp_1d . . . 230
15.77.2.32 mse_dp_2d . . . 231
15.77.2.33 mse_dp_3d . . . 231
15.77.2.34 mse_sp_1d . . . 232
15.77.2.35 mse_sp_2d . . . 232
15.77.2.36 mse_sp_3d . . . 233
15.77.2.37 nse_dp_1d . . . 233
15.77.2.38 nse_dp_2d . . . 234
15.77.2.39 nse_dp_3d . . . 234
15.77.2.40 nse_sp_1d . . . 235
15.77.2.41 nse_sp_2d . . . 235
15.77.2.42 nse_sp_3d . . . 235
15.77.2.43 rmse_dp_1d . . . 236
15.77.2.44 rmse_dp_2d . . . 236
15.77.2.45 rmse_dp_3d . . . 237
15.77.2.46 rmse_sp_1d . . . 237
15.77.2.47 rmse_sp_2d . . . 238
15.77.2.48 rmse_sp_3d . . . 238
15.77.2.49 sae_dp_1d . . . 239
15.77.2.50 sae_dp_2d . . . 239
15.77.2.51 sae_dp_3d . . . 240
15.77.2.52 sae_sp_1d . . . 240
15.77.2.53 sae_sp_2d . . . 241
15.77.2.54 sae_sp_3d . . . 241
15.77.2.55 sse_dp_1d . . . 242
15.77.2.56 sse_dp_2d . . . 242
15.77.2.57 sse_dp_3d . . . 243
15.77.2.58 sse_sp_1d . . . 243
15.77.2.59 sse_sp_2d . . . 244
15.77.2.60 sse_sp_3d . . . 244
15.77.2.61 wnse_dp_1d . . . 245
15.77.2.62 wnse_dp_2d . . . 245
15.77.2.63 wnse_dp_3d . . . 246
15.77.2.64 wnse_sp_1d . . . 246
15.77.2.65 wnse_sp_2d . . . 246
15.77.2.66 wnse_sp_3d . . . 247
15.78 mo_file Module Reference . . . 247
15.78.1 Detailed Description . . . 248

15.78.2 Member Data Documentation . . . 248
15.78.2.1 file_defoutput . . . 248
15.78.2.2 file_main . . . 248
15.78.2.3 file_namelist_mhm . . . 248
15.78.2.4 file_namelist_mhm_param . . . 249
15.78.2.5 udefoutput . . . 249
15.78.2.6 unamelist_mhm . . . 249
15.78.2.7 unamelist_mhm_param . . . 249
15.78.2.8 utws . . . 249
15.78.2.9 version . . . 249
15.78.2.10 version_date . . . 249
15.79 mo_finish Module Reference . . . 249
15.79.1 Detailed Description . . . 250
15.79.2 Member Function/Subroutine Documentation . . . 250
15.79.2.1 finish . . . 250
15.80 mo_global_variables Module Reference . . . 251
15.80.1 Detailed Description . . . 255
15.80.2 Member Data Documentation . . . 255
15.80.2.1 basin_avg_tws_obs . . . 255
15.80.2.2 basin_avg_tws_sim . . . 255
15.80.2.3 dirabsvappressure . . . 255
15.80.2.4 direvapotranspiration . . . 255
15.80.2.5 dirmaxtemperature . . . 255
15.80.2.6 dirmintemperature . . . 256
15.80.2.7 dirnetradiation . . . 256
15.80.2.8 dirneutrons . . . 256
15.80.2.9 dirprecipitation . . . 256
15.80.2.10 dirreferenceet . . . 256
15.80.2.11 dirsoil_moisture . . . 256
15.80.2.12 dirtemperature . . . 256
15.80.2.13 dirwindspeed . . . 256
15.80.2.14 evap_coeff . . . 256
15.80.2.15 fday_pet . . . 256
15.80.2.16 fday_prec . . . 256
15.80.2.17 fday_temp . . . 256
15.80.2.18 filetws . . . 257
15.80.2.19 fnight_pet . . . 257
15.80.2.20 fnight_prec . . . 257
15.80.2.21 fnight_temp . . . 257
15.80.2.22 inputformat_meteo_forcings . . . 257

15.80.2.23 1_absvappress . . . 257
15.80.2.24 1_aetcanopy . . . 257
15.80.2.25 1_aetsealed . . . 257
15.80.2.26 1_aetsoil . . . 257
15.80.2.27 1_baseflow . . . 257
15.80.2.28 1_et . . . 257
15.80.2.29 1_et_mask . . . 257
15.80.2.30 1_fastrunoff . . . 258
15.80.2.31 1_infilsoil . . . 258
15.80.2.32 1_inter . . . 258
15.80.2.33 1_melt . . . 258
15.80.2.34 1_netrad . . . 258
15.80.2.35 1_neutrons . . . 258
15.80.2.36 1_neutronsdata . . . 258
15.80.2.37 1_neutronsdata_mask . . . 258
15.80.2.38 1_percol . . . 258
15.80.2.39 1_pet . . . 258
15.80.2.40 1_pet_calc . . . 258
15.80.2.41 1_pet_weights . . . 258
15.80.2.42 1_pre . . . 259
15.80.2.43 1_pre_weights . . . 259
15.80.2.44 1_preeffect . . . 259
15.80.2.45 1_rain . . . 259
15.80.2.46 1_runoffseal . . . 259
15.80.2.47 1_satstw . . . 259
15.80.2.48 1_sealstw . . . 259
15.80.2.49 1_slowrunoff . . . 259
15.80.2.50 1_sm . . . 259
15.80.2.51 1_sm_mask . . . 259
15.80.2.52 1_snow . . . 259
15.80.2.53 1_snowpack . . . 259
15.80.2.54 1_soilmoist . . . 260
15.80.2.55 1_temp . . . 260
15.80.2.56 1_temp_weights . . . 260
15.80.2.57 1_throughfall . . . 260
15.80.2.58 1_tmax . . . 260
15.80.2.59 1_tmin . . . 260
15.80.2.60 1_total_runoff . . . 260
15.80.2.61 1_unsatstw . . . 260
15.80.2.62 1_windspeed . . . 260

15.80.2.63 level2 . . . 260
15.80.2.64 neutron_integral_afast . . . 260
15.80.2.65 nmeasperday_tws . . . 260
15.80.2.66 nsoilhorizons_sm_input . . . 261
15.80.2.67 ntimesteps_l1_et . . . 261
15.80.2.68 ntimesteps_l1_neutrons . . . 261
15.80.2.69 ntimesteps_l1_sm . . . 261
15.80.2.70 outputflxstate . . . 261
15.80.2.71 read_meteo_weights . . . 261
15.80.2.72 routingstates . . . 261
15.80.2.73 timestep_et_input . . . 261
15.80.2.74 timestep_model_inputs . . . 261
15.80.2.75 timestep_model_outputs . . . 261
15.80.2.76 timestep_neutrons_input . . . 261
15.80.2.77 timestep_sm_input . . . 261
15.81 mo_grid Module Reference . . . 262
15.81.1 Detailed Description . . . 262
15.81.2 Member Function/Subroutine Documentation . . . 262
15.81.2.1 calculate_grid_properties . . . 262
15.81.2.2 geocoordinates . . . 264
15.81.2.3 init_lowres_level . . . 265
15.81.2.4 l0_grid_setup . . . 266
15.81.2.5 mapcoordinates . . . 267
15.81.2.6 set_basin_indices . . . 267
15.82 mo_init_states Module Reference . . . 268
15.82.1 Detailed Description . . . 268
15.82.2 Member Function/Subroutine Documentation . . . 269
15.82.2.1 variables_alloc . . . 269
15.82.2.2 variables_default_init . . . 269
15.83 mo_julian Module Reference . . . 270
15.83.1 Detailed Description . . . 272
15.83.2 Member Function/Subroutine Documentation . . . 273
15.83.2.1 caldat . . . 273
15.83.2.2 caldat360 . . . 274
15.83.2.3 caldat365 . . . 275
15.83.2.4 caldatjulian . . . 276
15.83.2.5 date2dec . . . 277
15.83.2.6 date2dec360 . . . 279
15.83.2.7 date2dec365 . . . 279
15.83.2.8 date2decjulian . . . 280

15.83.2.9 dec2date . . . 281
15.83.2.10 dec2date360 . . . 283
15.83.2.11 dec2date365 . . . 284
15.83.2.12 dec2datejulian . . . 286
15.83.2.13 julday . . . 287
15.83.2.14 julday360 . . . 289
15.83.2.15 julday365 . . . 290
15.83.2.16 juldayjulian . . . 291
15.83.2.17 ndays . . . 292
15.83.2.18 ndyin . . . 293
15.83.2.19 selectcalendar . . . 294
15.83.2.20 setcalendarinteger . . . 295
15.83.2.21 setcalendarstring . . . 295
15.83.3 Member Data Documentation . . . 296
15.83.3.1 calendar . . . 296
15.84 mo_kind Module Reference . . . 296
15.84.1 Detailed Description . . . 298
15.84.2 Member Data Documentation . . . 298
15.84.2.1 dp . . . 298
15.84.2.2 dpc . . . 298
15.84.2.3 i1 . . . 298
15.84.2.4 i2 . . . 298
15.84.2.5 i4 . . . 298
15.84.2.6 i8 . . . 298
15.84.2.7 lgt . . . 299
15.84.2.8 sp . . . 299
15.84.2.9 spc . . . 299
15.85 mo_linfit Module Reference . . . 299
15.85.1 Detailed Description . . . 299
15.85.2 Member Function/Subroutine Documentation . . . 300
15.85.2.1 linfit_dp . . . 300
15.85.2.2 linfit_sp . . . 300
15.86 mo_mad Module Reference . . . 300
15.86.1 Detailed Description . . . 301
15.86.2 Member Function/Subroutine Documentation . . . 301
15.86.2.1 mad_dp . . . 301
15.86.2.2 mad_sp . . . 301
15.86.2.3 mad_val_dp . . . 302
15.86.2.4 mad_val_sp . . . 302
15.87 mo_mcmc Module Reference . . . 302

15.87.1 Detailed Description . . . 303
15.87.2 Member Function/Subroutine Documentation . . . 303
15.87.2.1 generatenewparameterset_dp . . . 303
15.87.2.2 mcmc_dp . . . 304
15.87.2.3 mcmc_stddev_dp . . . 304
15.87.2.4 pargen_dp . . . 305
15.87.2.5 pargennorm_dp . . . 305
15.88 mo_message Module Reference . . . 305
15.88.1 Detailed Description . . . 306
15.88.2 Member Function/Subroutine Documentation . . . 306
15.88.2.1 message . . . 306
15.88.3 Member Data Documentation . . . 307
15.88.3.1 message_text . . . 307
15.89 mo_meteo_forcings Module Reference . . . 307
15.89.1 Detailed Description . . . 308
15.89.2 Member Function/Subroutine Documentation . . . 308
15.89.2.1 chunk_config . . . 308
15.89.2.2 chunk_size . . . 309
15.89.2.3 is_read . . . 310
15.89.2.4 meteo_forcings_wrapper . . . 311
15.89.2.5 meteo_weights_wrapper . . . 312
15.89.2.6 prepare_meteo_forcings_data . . . 313
15.90 mo_mhm Module Reference . . . 314
15.90.1 Detailed Description . . . 315
15.90.2 Member Function/Subroutine Documentation . . . 316
15.90.2.1 mhm . . . 316
15.91 mo_mhm_constants Module Reference . . . 322
15.91.1 Detailed Description . . . 324
15.91.2 Member Data Documentation . . . 324
15.91.2.1 c1_initstatesm . . . 324
15.91.2.2 cosmic_alpha . . . 324
15.91.2.3 cosmic_bd . . . 324
15.91.2.4 cosmic_l1 . . . 324
15.91.2.5 cosmic_l2 . . . 324
15.91.2.6 cosmic_l3 . . . 324
15.91.2.7 cosmic_l4 . . . 325
15.91.2.8 cosmic_n . . . 325
15.91.2.9 cosmic_vwclat . . . 325
15.91.2.10 desilets_a0 . . . 325
15.91.2.11 desilets_a1 . . . 325

15.91.2.12 desilets_a2 . . . 325
15.91.2.13 duffiedelta1 . . . 325
15.91.2.14 duffiedelta2 . . . 325
15.91.2.15 duffiedr . . . 325
15.91.2.16 h2odens . . . 325
15.91.2.17 harsamconst . . . 325
15.91.2.18 noutflxstate . . . 325
15.91.2.19 p2_initstatefluxes . . . 326
15.91.2.20 p3_initstatefluxes . . . 326
15.91.2.21 p4_initstatefluxes . . . 326
15.91.2.22 p5_initstatefluxes . . . 326
15.91.2.23 satpressureslope1 . . . 326
15.91.2.24 stboltzmann . . . 326
15.91.2.25 tetens_c1 . . . 326
15.91.2.26 tetens_c2 . . . 326
15.91.2.27 tetens_c3 . . . 326
15.92 mo_mhm_eval Module Reference . . . 326
15.92.1 Detailed Description . . . 327
15.92.2 Member Function/Subroutine Documentation . . . 327
15.92.2.1 mhm_eval . . . 327
15.93 mo_mhm_read_config Module Reference . . . 330
15.93.1 Detailed Description . . . 330
15.93.2 Member Function/Subroutine Documentation . . . 330
15.93.2.1 mhm_read_config . . . 330
15.94 mo_moment Module Reference . . . 332
15.94.1 Detailed Description . . . 333
15.94.2 Member Function/Subroutine Documentation . . . 334
15.94.2.1 absdev_dp . . . 334
15.94.2.2 absdev_sp . . . 334
15.94.2.3 average_dp . . . 334
15.94.2.4 average_sp . . . 335
15.94.2.5 central_moment_dp . . . 335
15.94.2.6 central_moment_sp . . . 336
15.94.2.7 central_moment_var_dp . . . 336
15.94.2.8 central_moment_var_sp . . . 336
15.94.2.9 correlation_dp . . . 337
15.94.2.10 correlation_sp . . . 337
15.94.2.11 covariance_dp . . . 338
15.94.2.12 covariance_sp . . . 338
15.94.2.13 kurtosis_dp . . . 338

15.94.2.14 kurtosis_sp . . . 339
15.94.2.15 mean_dp . . . 339
15.94.2.16 mean_sp . . . 339
15.94.2.17 mixed_central_moment_dp . . . 340
15.94.2.18 mixed_central_moment_sp . . . 340
15.94.2.19 mixed_central_moment_var_dp . . . 341
15.94.2.20 mixed_central_moment_var_sp . . . 341
15.94.2.21 moment_dp . . . 341
15.94.2.22 moment_sp . . . 341
15.94.2.23 skewness_dp . . . 342
15.94.2.24 skewness_sp . . . 342
15.94.2.25 stddev_dp . . . 342
15.94.2.26 stddev_sp . . . 343
15.94.2.27 variance_dp . . . 343
15.94.2.28 variance_sp . . . 344
15.95 mo_mrm_constants Module Reference . . . 344
15.95.1 Detailed Description . . . 345
15.95.2 Member Data Documentation . . . 345
15.95.2.1 deltah . . . 345
15.95.2.2 given_ts . . . 345
15.95.2.3 maxnogauges . . . 345
15.95.2.4 noutflxstate . . . 345
15.95.2.5 nroutingstates . . . 345
15.95.2.6 rout_space_weight . . . 345
15.96 mo_mrm_eval Module Reference . . . 346
15.96.1 Detailed Description . . . 346
15.96.2 Member Function/Subroutine Documentation . . . 346
15.96.2.1 mrm_eval . . . 346
15.97 mo_mrm_file Module Reference . . . 348
15.97.1 Detailed Description . . . 350
15.97.2 Member Data Documentation . . . 350
15.97.2.1 file_config . . . 350
15.97.2.2 file_daily_discharge . . . 350
15.97.2.3 file_defoutput . . . 350
15.97.2.4 file_facc . . . 350
15.97.2.5 file_fdir . . . 350
15.97.2.6 file_gaugeloc . . . 350
15.97.2.7 file_gw_output . . . 350
15.97.2.8 file_main . . . 351
15.97.2.9 file_mrm_output . . . 351

15.97.2.10 file_namelist_mrm . . . 351
15.97.2.11 file_namelist_param_mrm . . . 351
15.97.2.12 file_slope . . . 351
15.97.2.13 ncfile_discharge . . . 351
15.97.2.14 uconfig . . . 351
15.97.2.15 udaily_discharge . . . 351
15.97.2.16 udefoutput . . . 351
15.97.2.17 udischarge . . . 352
15.97.2.18 ufacc . . . 352
15.97.2.19 ufdir . . . 352
15.97.2.20 ugaugeloc . . . 352
15.97.2.21 unamelist_mrm . . . 352
15.97.2.22 unamelist_param_mrm . . . 352
15.97.2.23 uslope . . . 352
15.97.2.24 version . . . 352
15.97.2.25 version_date . . . 352
15.98 mo_mrm_global_variables Module Reference . . . 353
15.98.1 Detailed Description . . . 356
15.98.2 Member Data Documentation . . . 356
15.98.2.1 basin_mrm . . . 356
15.98.2.2 dirbankfullrunoff . . . 356
15.98.2.3 dirgauges . . . 356
15.98.2.4 dirtotalrunoff . . . 356
15.98.2.5 filenametotalrunoff . . . 356
15.98.2.6 gauge . . . 357
15.98.2.7 gw_coupling . . . 357
15.98.2.8 inflowgauge . . . 357
15.98.2.9 is_start . . . 357
15.98.2.10 l0_celerity . . . 357
15.98.2.11 l0_channel_depth . . . 357
15.98.2.12 l0_channel_elevation . . . 357
15.98.2.13 l0_dracell . . . 357
15.98.2.14 l0_drasc . . . 357
15.98.2.15 l0_facc . . . 357
15.98.2.16 l0_fdir . . . 357
15.98.2.17 l0_floodplain . . . 357
15.98.2.18 l0_gaugeloc . . . 358
15.98.2.19 l0_inflowgaugeloc . . . 358
15.98.2.20 l0_l11_remap . . . 358
15.98.2.21 l0_noutlet . . . 358

15.98.2.22 0_river_head_mon_sum . . . 358
15.98.2.23 0_slope . . . 358
15.98.2.24 0_streamnet . . . 358
15.98.2.25 11_afloodplain . . . 358
15.98.2.26 11_areacell . . . 358
15.98.2.27 11_bankfull_runoff_in . . . 358
15.98.2.28 11_c1 . . . 358
15.98.2.29 11_c2 . . . 358
15.98.2.30 11_celerity . . . 359
15.98.2.31 11_cellcoor . . . 359
15.98.2.32 11_colout . . . 359
15.98.2.33 11_facc . . . 359
15.98.2.34 11_fcol . . . 359
15.98.2.35 11_fdir . . . 359
15.98.2.36 11_fromn . . . 359
15.98.2.37 11_frow . . . 359
15.98.2.38 11_k . . . 359
15.98.2.39 11_l1_id . . . 359
15.98.2.40 11_label . . . 359
15.98.2.41 11_length . . . 359
15.98.2.42 11_linkin_facc . . . 360
15.98.2.43 11_meandering . . . 360
15.98.2.44 11_netperm . . . 360
15.98.2.45 11_nlinkfracfpimp . . . 360
15.98.2.46 11_noutlets . . . 360
15.98.2.47 11_qmod . . . 360
15.98.2.48 11_qout . . . 360
15.98.2.49 11_qtin . . . 360
15.98.2.50 11_qtr . . . 360
15.98.2.51 11_rorder . . . 360
15.98.2.52 11_rowout . . . 360
15.98.2.53 11_sink . . . 360
15.98.2.54 11_slope . . . 361
15.98.2.55 11_tcol . . . 361
15.98.2.56 11_ton . . . 361
15.98.2.57 11_trow . . . 361
15.98.2.58 11_tsrout . . . 361
15.98.2.59 11_xi . . . 361
15.98.2.60 1_l11_id . . . 361
15.98.2.61 1_l11_remap . . . 361

15.98.2.62 l1_total_runoff_in . . . 361
15.98.2.63 level11 . . . 361
15.98.2.64 mrm_runoff . . . 361
15.98.2.65 ngaugestotal . . . 361
15.98.2.66 ninflowgaugestotal . . . 362
15.98.2.67 nmeasperday . . . 362
15.98.2.68 ntstepday . . . 362
15.98.2.69 outputflxstate_mrm . . . 362
15.98.2.70 timestep_model_outputs_mrm . . . 362
15.98.2.71 varnametotalrunoff . . . 362
15.99 mo_mrm_init Module Reference . . . 362
15.99.1 Detailed Description . . . 363
15.99.2 Member Function/Subroutine Documentation . . . 363
15.99.2.1 config_output . . . 363
15.99.2.2 l0_check_input_routing . . . 364
15.99.2.3 mrm_init . . . 365
15.99.2.4 print_startup_message . . . 367
15.99.2.5 variables_alloc_routing . . . 368
15.99.2.6 variables_default_init_routing . . . 368
15.100 mo_mrm_mpr Module Reference . . . 369
15.100.1 Detailed Description . . . 370
15.100.2 Member Function/Subroutine Documentation . . . 370
15.100.2.1 mrm_init_param . . . 370
15.100.2.2 mrm_update_param . . . 371
15.100.2.3 reg_rout . . . 372
15.101 mo_mrm_net_startup Module Reference . . . 373
15.101.1 Detailed Description . . . 374
15.101.2 Member Function/Subroutine Documentation . . . 374
15.101.2.1 celllength . . . 374
15.101.2.2 get_distance_two_lat_lon_points . . . 375
15.101.2.3 l11_calc_celerity . . . 376
15.101.2.4 l11_flow_accumulation . . . 377
15.101.2.5 l11_flow_direction . . . 378
15.101.2.6 l11_fraction_sealed_floodplain . . . 379
15.101.2.7 l11_l1_mapping . . . 380
15.101.2.8 l11_link_location . . . 380
15.101.2.9 l11_routing_order . . . 381
15.101.2.10 l01_set_drain_outlet_gauges . . . 382
15.101.2.11 l11_set_network_topology . . . 383
15.101.2.12 l21_stream_features . . . 384

15.101.2.13 movedownonecell . . . 385
15.101.2.14 moveup . . . 386
15.102 mo_mrm_objective_function_runoff Module Reference . . . 387
15.102.1 Detailed Description . . . 388
15.102.2 Member Function/Subroutine Documentation . . . 388
15.102.2.1 extract_runoff . . . 388
15.102.2.2 loglikelihood_evin2013_2 . . . 390
15.102.2.3 loglikelihood_stddev . . . 391
15.102.2.4 loglikelihood_trend_no_autocorr . . . 393
15.102.2.5 multi_objective_ae_fdc_lsv_nse_djf . . . 395
15.102.2.6 multi_objective_lnnse_highflow_lnnse_lowflow . . . 396
15.102.2.7 multi_objective_lnnse_highflow_lnnse_lowflow_2 . . . 397
15.102.2.8 multi_objective_nse_lnnse . . . 398
15.102.2.9 multi_objective_runoff . . . 400
15.102.2.10 objective_equal_nse_lnnse . . . 401
15.102.2.11 objective_kge . . . 402
15.102.2.12 objective_lnnse . . . 403
15.102.2.13 objective_multiple_gauges_kge_power6 . . . 405
15.102.2.14 objective_nse . . . 406
15.102.2.15 objective_power6_nse_lnnse . . . 407
15.102.2.16 objective_sse . . . 408
15.102.2.17 objective_weighted_nse . . . 409
15.102.2.18 parameter_regularization . . . 410
15.102.2.19 single_objective_runoff . . . 412
15.103 mo_mrm_read_config Module Reference . . . 414
15.103.1 Detailed Description . . . 414
15.103.2 Member Function/Subroutine Documentation . . . 414
15.103.2.1 mrm_read_config . . . 415
15.103.2.2 read_mrm_routing_params . . . 416
15.104 mo_mrm_read_data Module Reference . . . 417
15.104.1 Detailed Description . . . 418
15.104.2 Member Function/Subroutine Documentation . . . 418
15.104.2.1 mrm_read_bankfull_runoff . . . 418
15.104.2.2 mrm_read_discharge . . . 419
15.104.2.3 mrm_read_l0_data . . . 420
15.104.2.4 mrm_read_total_runoff . . . 421
15.104.2.5 rotate_fdir_variable . . . 422
15.105 mo_mrm_restart Module Reference . . . 422
15.105.1 Detailed Description . . . 423
15.105.2 Member Function/Subroutine Documentation . . . 423

15.105.2.1 mrm_read_restart_config . . . 423
15.105.2.2 mrm_read_restart_states . . . 424
15.105.2.3 mrm_write_restart . . . 425
15.106 mo_mrm_river_head Module Reference . . . 426
15.106.1 Detailed Description . . . 427
15.106.2 Member Function/Subroutine Documentation . . . 427
15.106.2.1 avg_and_write_timestep . . . 427
15.106.2.2 calc_channel_elevation . . . 427
15.106.2.3 calc_river_head . . . 428
15.106.2.4 calc_slope . . . 428
15.106.2.5 create_output . . . 429
15.106.2.6 init_masked_zeros_l0 . . . 430
15.106.2.7 reset_sum . . . 430
15.106.3 Member Data Documentation . . . 431
15.106.3.1 nc_riverhead . . . 431
15.106.3.2 nc_time . . . 431
15.106.3.3 sum_counter . . . 431
15.106.3.4 time_counter . . . 431
15.107 mo_mrm_routing Module Reference . . . 431
15.107.1 Detailed Description . . . 432
15.107.2 Member Function/Subroutine Documentation . . . 432
15.107.2.1 add_inflow . . . 432
15.107.2.2 l11_routing . . . 433
15.107.2.3 l11_runoff_acc . . . 435
15.107.2.4 mrm_routing . . . 436
15.108 mo_mrm_signatures Module Reference . . . 439
15.108.1 Detailed Description . . . 440
15.108.2 Member Function/Subroutine Documentation . . . 440
15.108.2.1 autocorrelation . . . 440
15.108.2.2 flowdurationcurve . . . 441
15.108.2.3 limb_densities . . . 443
15.108.2.4 maximummonthlyflow . . . 444
15.108.2.5 moments . . . 445
15.108.2.6 peakdistribution . . . 447
15.108.2.7 runoffratio . . . 448
15.108.2.8 zeroflowratio . . . 449
15.109 mo_mrm_write Module Reference . . . 450
15.109.1 Detailed Description . . . 451
15.109.2 Member Function/Subroutine Documentation . . . 451
15.109.2.1 mrm_write . . . 451

15.109.2.2 mrm_write_optifile . . . 452
15.109.2.3 mrm_write_optinamelist . . . 453
15.109.2.4 mrm_write_output_fluxes . . . 455
15.109.2.5 write_configfile . . . 456
15.109.2.6 write_daily_obs_sim_discharge . . . 457
15.109.3 Member Data Documentation . . . 458
15.109.3.1 average_counter . . . 458
15.109.3.2 day_counter . . . 458
15.109.3.3 month_counter . . . 458
15.109.3.4 nc . . . 458
15.109.3.5 year_counter . . . 458
15.110 mo_mrm_write_fluxes_states Module Reference . . . 458
15.110.1 Detailed Description . . . 459
15.110.2 Member Function/Subroutine Documentation . . . 459
15.110.2.1 close . . . 459
15.110.2.2 createoutputfile . . . 460
15.110.2.3 newoutputdataset . . . 461
15.110.2.4 newoutputvariable . . . 462
15.110.2.5 updatedataset . . . 463
15.110.2.6 updatevariable . . . 463
15.110.2.7 writetimestep . . . 464
15.110.2.8 writevariableattributes . . . 464
15.110.2.9 writevariabletimestep . . . 465
15.111 mo_ncread Module Reference . . . 465
15.111.1 Detailed Description . . . 466
15.111.2 Member Function/Subroutine Documentation . . . 466
15.111.2.1 check . . . 466
15.111.2.2 get_info . . . 467
15.111.2.3 get_ncdim . . . 469
15.111.2.4 get_ncdimatt . . . 471
15.111.2.5 get_ncvar_0d_dp . . . 472
15.111.2.6 get_ncvar_0d_i1 . . . 472
15.111.2.7 get_ncvar_0d_i4 . . . 472
15.111.2.8 get_ncvar_0d_sp . . . 473
15.111.2.9 get_ncvar_1d_dp . . . 473
15.111.2.10 get_ncvar_1d_i1 . . . 473
15.111.2.11 get_ncvar_1d_i4 . . . 474
15.111.2.12 get_ncvar_1d_sp . . . 474
15.111.2.13 get_ncvar_2d_dp . . . 474
15.111.2.14 get_ncvar_2d_i1 . . . 475

15.111.2.15 get_ncvar_2d_i4 . . . 475
15.111.2.16 get_ncvar_2d_sp . . . 475
15.111.2.17 get_ncvar_3d_dp . . . 476
15.111.2.18 get_ncvar_3d_i1 . . . 476
15.111.2.19 get_ncvar_3d_i4 . . . 476
15.111.2.20 get_ncvar_3d_sp . . . 477
15.111.2.21 get_ncvar_4d_dp . . . 477
15.111.2.22 get_ncvar_4d_i1 . . . 477
15.111.2.23 get_ncvar_4d_i4 . . . 478
15.111.2.24 get_ncvar_4d_sp . . . 478
15.111.2.25 get_ncvar_5d_dp . . . 478
15.111.2.26 get_ncvar_5d_i1 . . . 479
15.111.2.27 get_ncvar_5d_i4 . . . 479
15.111.2.28 get_ncvar_5d_sp . . . 480
15.111.2.29 get_ncvaratt . . . 480
15.111.2.30 ncclose . . . 480
15.111.2.31 ncopen . . . 481
15.112 mo_ncwrite Module Reference . . . 482
15.112.1 Detailed Description . . . 484
15.112.2 Member Function/Subroutine Documentation . . . 484
15.112.2.1 check . . . 484
15.112.2.2 close_netcdf . . . 484
15.112.2.3 create_netcdf . . . 485
15.112.2.4 dump_netcdf_1d_dp . . . 486
15.112.2.5 dump_netcdf_1d_i4 . . . 486
15.112.2.6 dump_netcdf_1d_sp . . . 487
15.112.2.7 dump_netcdf_2d_dp . . . 487
15.112.2.8 dump_netcdf_2d_i4 . . . 487
15.112.2.9 dump_netcdf_2d_sp . . . 488
15.112.2.10 dump_netcdf_3d_dp . . . 488
15.112.2.11 dump_netcdf_3d_i4 . . . 489
15.112.2.12 dump_netcdf_3d_sp . . . 489
15.112.2.13 dump_netcdf_4d_dp . . . 489
15.112.2.14 dump_netcdf_4d_i4 . . . 490
15.112.2.15 dump_netcdf_4d_sp . . . 490
15.112.2.16 dump_netcdf_5d_dp . . . 491
15.112.2.17 dump_netcdf_5d_i4 . . . 491
15.112.2.18 dump_netcdf_5d_sp . . . 491
15.112.2.19 open_netcdf . . . 492
15.112.2.20 var2nc_1d_dp . . . 494

15.112.2.21 var2nc_1d_i4 . . . 494
15.112.2.22 var2nc_1d_sp . . . 494
15.112.2.23 var2nc_2d_dp . . . 495
15.112.2.24 var2nc_2d_i4 . . . 495
15.112.2.25 var2nc_2d_sp . . . 496
15.112.2.26 var2nc_3d_dp . . . 496
15.112.2.27 var2nc_3d_i4 . . . 497
15.112.2.28 var2nc_3d_sp . . . 497
15.112.2.29 var2nc_4d_dp . . . 498
15.112.2.30 var2nc_4d_i4 . . . 498
15.112.2.31 var2nc_4d_sp . . . 499
15.112.2.32 var2nc_5d_dp . . . 499
15.112.2.33 var2nc_5d_i4 . . . 500
15.112.2.34 var2nc_5d_sp . . . 500
15.112.2.35 write_dynamic_netcdf . . . 501
15.112.2.36 write_static_netcdf . . . 501
15.112.3 Member Data Documentation . . . 501
15.112.3.1 dnc . . . 502
15.112.3.2 gatt . . . 502
15.112.3.3 maxlen . . . 502
15.112.3.4 nattdim . . . 502
15.112.3.5 ndims . . . 502
15.112.3.6 ngatt . . . 502
15.112.3.7 nmaxatt . . . 502
15.112.3.8 nmaxdim . . . 502
15.112.3.9 nvars . . . 502
15.112.3.10 v . . . 502
15.113 mo_netcdf Module Reference . . . 502
15.113.1 Detailed Description . . . 506
15.113.2 Member Function/Subroutine Documentation . . . 506
15.113.2.1 check . . . 506
15.113.2.2 close . . . 506
15.113.2.3 equalncdimensions . . . 506
15.113.2.4 getdata1df32 . . . 506
15.113.2.5 getdata1df64 . . . 507
15.113.2.6 getdata1di16 . . . 507
15.113.2.7 getdata1di32 . . . 508
15.113.2.8 getdata1di64 . . . 508
15.113.2.9 getdata1di8 . . . 509
15.113.2.10 getdata2df32 . . . 509

15.113.2.11 getdata2df64 . . . 510
15.113.2.12 getdata2di16 . . . 510
15.113.2.13 getdata2di32 . . . 511
15.113.2.14 getdata2di64 . . . 511
15.113.2.15 getdata2di8 . . . 512
15.113.2.16 getdata3df32 . . . 512
15.113.2.17 getdata3df64 . . . 513
15.113.2.18 getdata3di16 . . . 513
15.113.2.19 getdata3di32 . . . 514
15.113.2.20 getdata3di64 . . . 514
15.113.2.21 getdata3di8 . . . 515
15.113.2.22 getdata4df32 . . . 515
15.113.2.23 getdata4df64 . . . 516
15.113.2.24 getdata4di16 . . . 516
15.113.2.25 getdata4di32 . . . 517
15.113.2.26 getdata4di64 . . . 517
15.113.2.27 getdata4di8 . . . 518
15.113.2.28 getdata5df32 . . . 518
15.113.2.29 getdata5df64 . . . 519
15.113.2.30 getdata5di16 . . . 519
15.113.2.31 getdata5di32 . . . 520
15.113.2.32 getdata5di64 . . . 520
15.113.2.33 getdata5di8 . . . 521
15.113.2.34 getdatascalarf32 . . . 521
15.113.2.35 getdatascalarf64 . . . 522
15.113.2.36 getdatascalari16 . . . 522
15.113.2.37 getdatascalari32 . . . 522
15.113.2.38 getdatascalari64 . . . 523
15.113.2.39 getdatascalari8 . . . 523
15.113.2.40 getdimensionbyid . . . 523
15.113.2.41 getdimensionbyname . . . 524
15.113.2.42 getdimensionlength . . . 524
15.113.2.43 getdimensionname . . . 525
15.113.2.44 getdtypefrominteger . . . 525
15.113.2.45 getdtypefromstring . . . 525
15.113.2.46 getglobalattributechar . . . 525
15.113.2.47 getglobalattributef32 . . . 526
15.113.2.48 getglobalattributef64 . . . 526
15.113.2.49 getglobalattributei16 . . . 526
15.113.2.50 getglobalattributei32 . . . 527

15.113.2.51 getglobalattributei64 . . . 527
15.113.2.52 getglobalattributei8 . . . 527
15.113.2.53 getnodimensions . . . 528
15.113.2.54 getnovariables . . . 528
15.113.2.55 getreaddatashape . . . 529
15.113.2.56 getunlimiteddimension . . . 530
15.113.2.57 getvariableattributechar . . . 531
15.113.2.58 getvariableattributef32 . . . 531
15.113.2.59 getvariableattributef64 . . . 532
15.113.2.60 getvariableattributei16 . . . 532
15.113.2.61 getvariableattributei32 . . . 532
15.113.2.62 getvariableattributei64 . . . 533
15.113.2.63 getvariableattributei8 . . . 533
15.113.2.64 getvariablebyname . . . 533
15.113.2.65 getvariabledimensions . . . 534
15.113.2.66 getvariabledtype . . . 534
15.113.2.67 getvariablefillvaluef32 . . . 535
15.113.2.68 getvariablefillvaluef64 . . . 535
15.113.2.69 getvariablefillvaluei16 . . . 535
15.113.2.70 getvariablefillvaluei32 . . . 535
15.113.2.71 getvariablefillvaluei64 . . . 535
15.113.2.72 getvariablefillvaluei8 . . . 535
15.113.2.73 getvariableids . . . 536
15.113.2.74 getvariablename . . . 536
15.113.2.75 getvariables . . . 537
15.113.2.76 getvariableshape . . . 537
15.113.2.77 hasattribute . . . 537
15.113.2.78 hasdimension . . . 538
15.113.2.79 hasvariable . . . 538
15.113.2.80 initncdataset . . . 538
15.113.2.81 initncdimension . . . 539
15.113.2.82 initncvariable . . . 539
15.113.2.83 isdatasetunlimited . . . 539
15.113.2.84 isunlimiteddimension . . . 539
15.113.2.85 isunlimitedvariable . . . 539
15.113.2.86 newncdataset . . . 540
15.113.2.87 newncdimension . . . 540
15.113.2.88 newncvariable . . . 540
15.113.2.89 setdata1df32 . . . 540
15.113.2.90 setdata1df64 . . . 541

15.113.2.91 setdata1di16 . . . 541
15.113.2.92 setdata1di32 . . . 542
15.113.2.93 setdata1di64 . . . 542
15.113.2.94 setdata1di8 . . . 542
15.113.2.95 setdata2df32 . . . 543
15.113.2.96 setdata2df64 . . . 543
15.113.2.97 setdata2di16 . . . 544
15.113.2.98 setdata2di32 . . . 544
15.113.2.99 setdata2di64 . . . 544
15.113.2.100 setdata2di8 . . . 545
15.113.2.101 setdata3df32 . . . 545
15.113.2.102 setdata3df64 . . . 546
15.113.2.103 setdata3di16 . . . 546
15.113.2.104 setdata3di32 . . . 546
15.113.2.105 setdata3di64 . . . 547
15.113.2.106 setdata3di8 . . . 547
15.113.2.107 setdata4df32 . . . 548
15.113.2.108 setdata4df64 . . . 548
15.113.2.109 setdata4di16 . . . 548
15.113.2.110 setdata4di32 . . . 549
15.113.2.111 setdata4di64 . . . 549
15.113.2.112 setdata4di8 . . . 550
15.113.2.113 setdata5df32 . . . 550
15.113.2.114 setdata5df64 . . . 550
15.113.2.115 setdata5di16 . . . 551
15.113.2.116 setdata5di32 . . . 551
15.113.2.117 setdata5di64 . . . 552
15.113.2.118 setdata5di8 . . . 552
15.113.2.119 setdatascalarf32 . . . 552
15.113.2.120 setdatascalarf64 . . . 553
15.113.2.121 setdatascalari16 . . . 553
15.113.2.122 setdatascalari32 . . . 553
15.113.2.123 setdatascalari64 . . . 554
15.113.2.124 setdatascalari8 . . . 554
15.113.2.125 setdimension . . . 555
15.113.2.126 setglobalattributechar . . . 555
15.113.2.127 setglobalattributef32 . . . 555
15.113.2.128 setglobalattributef64 . . . 556
15.113.2.129 setglobalattributei16 . . . 556
15.113.2.130 setglobalattributei32 . . . 556

15.113.2.31 setglobalattributei64 . . . 557
15.113.2.32 setglobalattributei8 . . . 557
15.113.2.33 setvariableattributechar . . . 557
15.113.2.34 setvariableattributef32 . . . 558
15.113.2.35 setvariableattributef64 . . . 558
15.113.2.36 setvariableattributei16 . . . 558
15.113.2.37 setvariableattributei32 . . . 559
15.113.2.38 setvariableattributei64 . . . 559
15.113.2.39 setvariableattributei8 . . . 559
15.113.2.40 setvariablefillvaluef32 . . . 560
15.113.2.41 setvariablefillvaluef64 . . . 560
15.113.2.42 setvariablefillvaluei16 . . . 560
15.113.2.43 setvariablefillvaluei32 . . . 560
15.113.2.44 setvariablefillvaluei64 . . . 560
15.113.2.45 setvariablefillvaluei8 . . . 560
15.113.2.46 setvariablewithids . . . 560
15.113.2.47 setvariablewithnames . . . 561
15.113.2.48 setvariablewithtypes . . . 561
15.114 mo_neutrons Module Reference . . . 562
15.114.1 Detailed Description . . . 563
15.114.2 Member Function/Subroutine Documentation . . . 563
15.114.2.1 approx_mon_int . . . 563
15.114.2.2 approx_mon_int_eps . . . 564
15.114.2.3 approx_mon_int_steps . . . 565
15.114.2.4 cosmic . . . 565
15.114.2.5 desiletsn0 . . . 567
15.114.2.6 intgrandfast . . . 568
15.114.2.7 lookupintegral . . . 569
15.114.2.8 oldintegration . . . 569
15.114.2.9 tabularintegralafast . . . 570
15.115 mo_nml Module Reference . . . 571
15.115.1 Detailed Description . . . 572
15.115.2 Member Function/Subroutine Documentation . . . 572
15.115.2.1 close_nml . . . 572
15.115.2.2 open_nml . . . 573
15.115.2.3 position_nml . . . 574
15.115.3 Member Data Documentation . . . 575
15.115.3.1 length_error . . . 575
15.115.3.2 missing . . . 575
15.115.3.3 nunitnml . . . 575

15.115.3.4 positioned . . . 576
15.115.3.5 read_error . . . 576
15.116 mo_objective_function Module Reference . . . 576
15.116.1 Detailed Description . . . 577
15.116.2 Member Function/Subroutine Documentation . . . 577
15.116.2.1 extract_basin_avg_tws . . . 577
15.116.2.2 objective . . . 578
15.116.2.3 objective_et_kge_catchment_avg . . . 580
15.116.2.4 objective_kge_q_et . . . 581
15.116.2.5 objective_kge_q_rmse_et . . . 582
15.116.2.6 objective_kge_q_rmse_tws . . . 583
15.116.2.7 objective_kge_q_sm_corr . . . 584
15.116.2.8 objective_neutrons_kge_catchment_avg . . . 586
15.116.2.9 objective_sm_corr . . . 588
15.116.2.10 objective_sm_kge_catchment_avg . . . 589
15.116.2.11 objective_sm_pd . . . 590
15.116.2.12 objective_sm_sse_standard_score . . . 592
15.117 mo_optimization_utils Module Reference . . . 593
15.117.1 Detailed Description . . . 593
15.118 mo_orderpack Module Reference . . . 593
15.118.1 Detailed Description . . . 596
15.118.2 Member Function/Subroutine Documentation . . . 596
15.118.2.1 d_ctrper . . . 596
15.118.2.2 d_fndnth . . . 596
15.118.2.3 d_indmed . . . 597
15.118.2.4 d_indnth . . . 597
15.118.2.5 d_inspar . . . 597
15.118.2.6 d_inssor . . . 597
15.118.2.7 d_med . . . 597
15.118.2.8 d_median . . . 598
15.118.2.9 d_mrgref . . . 598
15.118.2.10 d_mrgrnk . . . 598
15.118.2.11 d_mulcnt . . . 598
15.118.2.12 d_nearless . . . 598
15.118.2.13 d_rapknr . . . 598
15.118.2.14 d_refpar . . . 598
15.118.2.15 d_refsor . . . 598
15.118.2.16 d_rinpar . . . 599
15.118.2.17 d_rnkpar . . . 599
15.118.2.18 d_subsor . . . 599

15.118.2.19_uniinv . . . 599
15.118.2.20_unipar . . . 600
15.118.2.21_unirnk . . . 600
15.118.2.22_unista . . . 600
15.118.2.23_valmed . . . 600
15.118.2.24_valnth . . . 600
15.118.2.25ctrper . . . 600
15.118.2.26fndnth . . . 600
15.118.2.27indmed . . . 601
15.118.2.28indnth . . . 601
15.118.2.29inspar . . . 601
15.118.2.30inssor . . . 601
15.118.2.31med . . . 601
15.118.2.32median . . . 602
15.118.2.33mrgref . . . 602
15.118.2.34mrgrnk . . . 602
15.118.2.35mulcnt . . . 602
15.118.2.36nearless . . . 602
15.118.2.37rapknr . . . 602
15.118.2.38refpar . . . 602
15.118.2.39refsor . . . 602
15.118.2.40rinpar . . . 603
15.118.2.41rnkpar . . . 603
15.118.2.42subsor . . . 603
15.118.2.43uniinv . . . 603
15.118.2.44unipar . . . 604
15.118.2.45unirnk . . . 604
15.118.2.46unista . . . 604
15.118.2.47valmed . . . 604
15.118.2.48valnth . . . 604
15.118.2.49_ctrper . . . 604
15.118.2.50_fndnth . . . 604
15.118.2.51_indmed . . . 605
15.118.2.52_indnth . . . 605
15.118.2.53_inspar . . . 605
15.118.2.54_inssor . . . 605
15.118.2.55_med . . . 605
15.118.2.56_median . . . 606
15.118.2.57_mrgref . . . 606
15.118.2.58_mrgrnk . . . 606

15.118.2.59 mulcnt . . . 606
15.118.2.60 nearless . . . 606
15.118.2.61 rapknr . . . 606
15.118.2.62 refpar . . . 606
15.118.2.63 refsor . . . 606
15.118.2.64 rinpar . . . 607
15.118.2.65 rnkpar . . . 607
15.118.2.66 subsor . . . 607
15.118.2.67 uniinv . . . 607
15.118.2.68 unipar . . . 608
15.118.2.69 unirnk . . . 608
15.118.2.70 unista . . . 608
15.118.2.71 valmed . . . 608
15.118.2.72 valnth . . . 608
15.118.2.73 sort_index_dp . . . 608
15.118.2.74 sort_index_i4 . . . 609
15.118.2.75 sort_index_sp . . . 609
15.118.3 Member Data Documentation . . . 609
15.118.3.1 idont . . . 610
15.119 mo_percentile Module Reference . . . 610
15.119.1 Detailed Description . . . 610
15.119.2 Member Function/Subroutine Documentation . . . 610
15.119.2.1 median_dp . . . 610
15.119.2.2 median_sp . . . 611
15.119.2.3 n_element_dp . . . 611
15.119.2.4 n_element_sp . . . 611
15.119.2.5 percentile_0d_dp . . . 612
15.119.2.6 percentile_0d_sp . . . 612
15.119.2.7 percentile_1d_dp . . . 613
15.119.2.8 percentile_1d_sp . . . 613
15.119.2.9 qmedian_dp . . . 613
15.119.2.10 qmedian_sp . . . 614
15.120 mo_pet Module Reference . . . 614
15.120.1 Detailed Description . . . 615
15.120.2 Member Function/Subroutine Documentation . . . 615
15.120.2.1 extraterr_rad_approx . . . 615
15.120.2.2 pet_hargreaves . . . 616
15.120.2.3 pet_penman . . . 617
15.120.2.4 pet_priestly . . . 619
15.120.2.5 sat_vap_pressure . . . 620

15.120.2.6 slope_satpressure . . . 621
15.121 mo_read_forcing_nc Module Reference . . . 622
15.121.1 Detailed Description . . . 622
15.121.2 Member Function/Subroutine Documentation . . . 622
15.121.2.1 get_time_vector_and_select . . . 622
15.121.2.2 read_const_forcing_nc . . . 624
15.121.2.3 read_forcing_nc . . . 625
15.121.2.4 read_weights_nc . . . 626
15.122 mo_read_latlon Module Reference . . . 629
15.122.1 Detailed Description . . . 629
15.122.2 Member Function/Subroutine Documentation . . . 629
15.122.2.1 read_latlon . . . 629
15.123 mo_read_optional_data Module Reference . . . 631
15.123.1 Detailed Description . . . 631
15.123.2 Member Function/Subroutine Documentation . . . 631
15.123.2.1 read_basin_avg_tws . . . 631
15.123.2.2 read_evapotranspiration . . . 632
15.123.2.3 read_neutrons . . . 633
15.123.2.4 read_soil_moisture . . . 634
15.124 mo_read_spatial_data Module Reference . . . 635
15.124.1 Detailed Description . . . 636
15.124.2 Member Function/Subroutine Documentation . . . 636
15.124.2.1 read_header_ascii . . . 636
15.124.2.2 read_spatial_data_ascii_dp . . . 637
15.124.2.3 read_spatial_data_ascii_i4 . . . 638
15.125 mo_read_timeseries Module Reference . . . 639
15.125.1 Detailed Description . . . 639
15.125.2 Member Function/Subroutine Documentation . . . 640
15.125.2.1 read_timeseries . . . 640
15.126 mo_restart Module Reference . . . 641
15.126.1 Detailed Description . . . 642
15.126.2 Member Function/Subroutine Documentation . . . 642
15.126.2.1 read_restart_states . . . 642
15.126.2.2 unpack_field_and_write_1d_dp . . . 643
15.126.2.3 unpack_field_and_write_1d_i4 . . . 643
15.126.2.4 unpack_field_and_write_2d_dp . . . 644
15.126.2.5 unpack_field_and_write_3d_dp . . . 644
15.126.2.6 write_restart_files . . . 644
15.127 mo_runoff Module Reference . . . 645
15.127.1 Detailed Description . . . 645

15.127.2 Member Function/Subroutine Documentation . . . 645
15.127.2.1 l1_total_runoff . . . 646
15.127.2.2 runoff_sat_zone . . . 647
15.127.2.3 runoff_unsat_zone . . . 648
15.128 mo_sce Module Reference . . . 649
15.128.1 Detailed Description . . . 649
15.128.2 Member Function/Subroutine Documentation . . . 650
15.128.2.1 cce . . . 650
15.128.2.2 chkcst . . . 650
15.128.2.3 comp . . . 651
15.128.2.4 getpnt . . . 651
15.128.2.5 parstt . . . 652
15.128.2.6 sce . . . 652
15.128.2.7 sort_matrix . . . 656
15.129 mo_set_netcdf_outputs Module Reference . . . 656
15.129.1 Detailed Description . . . 657
15.129.2 Member Function/Subroutine Documentation . . . 657
15.129.2.1 set_netcdf . . . 657
15.130 mo_snow_accum_melt Module Reference . . . 657
15.130.1 Detailed Description . . . 658
15.130.2 Member Function/Subroutine Documentation . . . 658
15.130.2.1 snow_accum_melt . . . 658
15.131 mo_soil_moisture Module Reference . . . 659
15.131.1 Detailed Description . . . 660
15.131.2 Member Function/Subroutine Documentation . . . 660
15.131.2.1 feddes_et_reduction . . . 660
15.131.2.2 jarvis_et_reduction . . . 661
15.131.2.3 soil_moisture . . . 662
15.132 mo_spatial_agg_disagg_forcing Module Reference . . . 664
15.132.1 Detailed Description . . . 665
15.132.2 Member Function/Subroutine Documentation . . . 665
15.132.2.1 spatial_aggregation_3d . . . 665
15.132.2.2 spatial_aggregation_4d . . . 665
15.132.2.3 spatial_disaggregation_3d . . . 665
15.132.2.4 spatial_disaggregation_4d . . . 665
15.133 mo_spatialsimilarity Module Reference . . . 665
15.133.1 Detailed Description . . . 666
15.133.2 Member Function/Subroutine Documentation . . . 666
15.133.2.1 nndv_dp . . . 666
15.133.2.2 nndv_sp . . . 667

15.133.2.3 pd_dp . . . 667
15.133.2.4 pd_sp . . . 667
15.134 mo_standard_score Module Reference . . . 668
15.134.1 Detailed Description . . . 668
15.134.2 Member Function/Subroutine Documentation . . . 669
15.134.2.1 classified_standard_score_dp . . . 669
15.134.2.2 classified_standard_score_sp . . . 669
15.134.2.3 standard_score_dp . . . 670
15.134.2.4 standard_score_sp . . . 670
15.135 mo_startup Module Reference . . . 670
15.135.1 Detailed Description . . . 671
15.135.2 Member Function/Subroutine Documentation . . . 671
15.135.2.1 constants_init . . . 671
15.135.2.2 i2_variable_init . . . 672
15.135.2.3 mhm_initialize . . . 673
15.136 mo_string_utils Module Reference . . . 674
15.136.1 Detailed Description . . . 675
15.136.2 Member Function/Subroutine Documentation . . . 675
15.136.2.1 compress . . . 675
15.136.2.2 divide_string . . . 676
15.136.2.3 dp2str . . . 677
15.136.2.4 equalstrings . . . 677
15.136.2.5 i42str . . . 678
15.136.2.6 i64array2str . . . 678
15.136.2.7 i82str . . . 678
15.136.2.8 log2str . . . 678
15.136.2.9 nonull . . . 679
15.136.2.10 sp2str . . . 679
15.136.2.11 splitstring . . . 680
15.136.2.12 startswith . . . 680
15.136.2.13 str2num . . . 681
15.136.2.14 tolower . . . 681
15.136.2.15 toupper . . . 682
15.136.3 Member Data Documentation . . . 682
15.136.3.1 separator . . . 682
15.137 mo_template Module Reference . . . 682
15.137.1 Detailed Description . . . 683
15.137.2 Member Function/Subroutine Documentation . . . 684
15.137.2.1 circum . . . 684
15.137.2.2 mean_dp . . . 684

15.137.2.3 mean_sp . . . 685
15.137.3 Member Data Documentation . . . 685
15.137.3.1 itest . . . 685
15.137.3.2 pi_dp . . . 685
15.137.3.3 pi_sp . . . 685
15.138 mo_temporal_aggregation Module Reference . . . 685
15.138.1 Detailed Description . . . 686
15.138.2 Member Function/Subroutine Documentation . . . 686
15.138.2.1 day2mon_average_dp . . . 686
15.138.2.2 hour2day_average_dp . . . 686
15.139 mo_temporal_disagg_forcing Module Reference . . . 687
15.139.1 Detailed Description . . . 687
15.139.2 Member Function/Subroutine Documentation . . . 687
15.139.2.1 temporal_disagg_forcing . . . 688
15.140 mo_timer Module Reference . . . 689
15.140.1 Detailed Description . . . 690
15.140.2 Member Function/Subroutine Documentation . . . 690
15.140.2.1 timer_check . . . 690
15.140.2.2 timer_clear . . . 691
15.140.2.3 timer_get . . . 691
15.140.2.4 timer_print . . . 692
15.140.2.5 timer_start . . . 694
15.140.2.6 timer_stop . . . 695
15.140.2.7 timers_init . . . 696
15.140.3 Member Data Documentation . . . 697
15.140.3.1 clock_rate . . . 697
15.140.3.2 cputime . . . 697
15.140.3.3 cycles1 . . . 697
15.140.3.4 cycles2 . . . 697
15.140.3.5 cycles_max . . . 697
15.140.3.6 max_timers . . . 698
15.140.3.7 status . . . 698
15.141 mo_utils Module Reference . . . 698
15.141.1 Detailed Description . . . 699
15.141.2 Member Function/Subroutine Documentation . . . 699
15.141.2.1 equal_dp . . . 699
15.141.2.2 equal_sp . . . 700
15.141.2.3 greaterequal_dp . . . 700
15.141.2.4 greaterequal_sp . . . 700
15.141.2.5 is_finite_dp . . . 701

15.141.2.6 is_finite_sp . . . 701
15.141.2.7 is_nan_dp . . . 701
15.141.2.8 is_nan_sp . . . 702
15.141.2.9 is_normal_dp . . . 702
15.141.2.10 is_normal_sp . . . 702
15.141.2.11 lesserequal_dp . . . 703
15.141.2.12 lesserequal_sp . . . 703
15.141.2.13 locate_0d_dp . . . 703
15.141.2.14 locate_0d_sp . . . 704
15.141.2.15 locate_1d_dp . . . 704
15.141.2.16 locate_1d_sp . . . 705
15.141.2.17 notequal_dp . . . 705
15.141.2.18 notequal_sp . . . 705
15.141.2.19 special_value_dp . . . 706
15.141.2.20 special_value_sp . . . 706
15.141.2.21 swap_vec_dp . . . 707
15.141.2.22 swap_vec_i4 . . . 707
15.141.2.23 swap_vec_sp . . . 707
15.141.2.24 swap_xy_dp . . . 707
15.141.2.25 swap_xy_i4 . . . 707
15.141.2.26 swap_xy_sp . . . 707
15.142 mo_write_ascii Module Reference . . . 708
15.142.1 Detailed Description . . . 708
15.142.2 Member Function/Subroutine Documentation . . . 708
15.142.2.1 write_configfile . . . 708
15.142.2.2 write_optifile . . . 709
15.142.2.3 write_optinamelist . . . 710
15.143 mo_write_fluxes_states Module Reference . . . 712
15.143.1 Detailed Description . . . 712
15.143.2 Member Function/Subroutine Documentation . . . 713
15.143.2.1 close . . . 713
15.143.2.2 createoutputfile . . . 713
15.143.2.3 fluxesunit . . . 714
15.143.2.4 newoutputdataset . . . 715
15.143.2.5 newoutputvariable . . . 716
15.143.2.6 updatedataset . . . 717
15.143.2.7 updatevariable . . . 719
15.143.2.8 writetimestep . . . 720
15.143.2.9 writevariableattributes . . . 720
15.143.2.10 writevariabletimestep . . . 721

15.144 mo_xor4096 Module Reference . . . 722
15.144.1 Detailed Description . . . 723
15.144.2 Member Function/Subroutine Documentation . . . 723
15.144.2.1 get_timeseed_i4_0d . . . 723
15.144.2.2 get_timeseed_i4_1d . . . 723
15.144.2.3 get_timeseed_i8_0d . . . 723
15.144.2.4 get_timeseed_i8_1d . . . 723
15.144.2.5 xor4096d_0d . . . 723
15.144.2.6 xor4096d_1d . . . 723
15.144.2.7 xor4096f_0d . . . 723
15.144.2.8 xor4096f_1d . . . 723
15.144.2.9 xor4096gd_0d . . . 724
15.144.2.10 xor4096gd_1d . . . 724
15.144.2.11 xor4096gf_0d . . . 724
15.144.2.12 xor4096gf_1d . . . 724
15.144.2.13 xor4096l_0d . . . 724
15.144.2.14 xor4096l_1d . . . 724
15.144.2.15 xor4096s_0d . . . 724
15.144.2.16 xor4096s_1d . . . 724
15.144.3 Member Data Documentation . . . 724
15.144.3.1 n_save_state . . . 724
15.145 mo_moment::moment Interface Reference . . . 725
15.145.1 Detailed Description . . . 725
15.145.2 Member Function/Subroutine Documentation . . . 725
15.145.2.1 moment_dp . . . 725
15.145.2.2 moment_sp . . . 725
15.146 mo_orderpack::mrgref Interface Reference . . . 725
15.146.1 Detailed Description . . . 725
15.146.2 Member Function/Subroutine Documentation . . . 725
15.146.2.1 d_mrgref . . . 725
15.146.2.2 i_mrgref . . . 726
15.146.2.3 r_mrgref . . . 726
15.147 mo_orderpack::mrgrnk Interface Reference . . . 726
15.147.1 Detailed Description . . . 726
15.147.2 Member Function/Subroutine Documentation . . . 726
15.147.2.1 d_mrgrnk . . . 726
15.147.2.2 i_mrgrnk . . . 726
15.147.2.3 r_mrgrnk . . . 726
15.148 mo_errormeasures::mse Interface Reference . . . 726
15.148.1 Detailed Description . . . 727

15.148.2 Member Function/Subroutine Documentation . . . 727
15.148.2.1 mse_dp_1d . . . 727
15.148.2.2 mse_dp_2d . . . 727
15.148.2.3 mse_dp_3d . . . 728
15.148.2.4 mse_sp_1d . . . 728
15.148.2.5 mse_sp_2d . . . 729
15.148.2.6 mse_sp_3d . . . 729
15.149 mo_orderpack::mulcnt Interface Reference . . . 730
15.149.1 Detailed Description . . . 730
15.149.2 Member Function/Subroutine Documentation . . . 730
15.149.2.1 d_mulcnt . . . 730
15.149.2.2 i_mulcnt . . . 730
15.149.2.3 r_mulcnt . . . 730
15.150 mo_percentile::n_element Interface Reference . . . 731
15.150.1 Detailed Description . . . 731
15.150.2 Member Function/Subroutine Documentation . . . 731
15.150.2.1 n_element_dp . . . 731
15.150.2.2 n_element_sp . . . 731
15.151 mo_netcdf::ncdataset Interface Reference . . . 732
15.151.1 Detailed Description . . . 733
15.151.2 Member Function/Subroutine Documentation . . . 734
15.151.2.1 close . . . 734
15.151.2.2 getattribute . . . 734
15.151.2.3 getdimension . . . 735
15.151.2.4 getdimensionbyid . . . 736
15.151.2.5 getdimensionbyname . . . 737
15.151.2.6 getglobalattributechar . . . 737
15.151.2.7 getglobalattributef32 . . . 738
15.151.2.8 getglobalattributef64 . . . 738
15.151.2.9 getglobalattributei16 . . . 738
15.151.2.10 getglobalattributei32 . . . 738
15.151.2.11 getglobalattributei64 . . . 739
15.151.2.12 getglobalattributei8 . . . 739
15.151.2.13 getnovariables . . . 739
15.151.2.14 getunlimiteddimension . . . 740
15.151.2.15 getvariable . . . 740
15.151.2.16 getvariablebyname . . . 741
15.151.2.17 getvariableids . . . 742
15.151.2.18 getvariables . . . 742
15.151.2.19 hasdimension . . . 742

15.151.2.20 masvariable . . . 743
15.151.2.21 initncdataset . . . 744
15.151.2.22 isunlimited . . . 744
15.151.2.23 setattribute . . . 744
15.151.2.24 setdimension . . . 746
15.151.2.25 setglobalattributechar . . . 747
15.151.2.26 setglobalattributef32 . . . 747
15.151.2.27 setglobalattributef64 . . . 747
15.151.2.28 setglobalattributei16 . . . 747
15.151.2.29 setglobalattributei32 . . . 748
15.151.2.30 setglobalattributei64 . . . 748
15.151.2.31 setglobalattributei8 . . . 748
15.151.2.32 setvariable . . . 749
15.151.2.33 setvariablewithids . . . 750
15.151.2.34 setvariablewithnames . . . 751
15.151.2.35 setvariablewithtypes . . . 751
15.151.3 Member Data Documentation . . . 751
15.151.3.1 dataset . . . 751
15.151.3.2 file . . . 751
15.151.3.3 filename . . . 751
15.151.3.4 fname . . . 752
15.151.3.5 id . . . 752
15.151.3.6 mode . . . 752
15.151.3.7 netcdf . . . 752
15.151.3.8 of . . . 752
15.151.3.9 open . . . 752
15.151.3.10 opened . . . 752
15.151.3.11 the . . . 752
15.152 mo_netcdf::ncdimension Type Reference . . . 752
15.152.1 Detailed Description . . . 756
15.152.2 Member Function/Subroutine Documentation . . . 756
15.152.2.1 getattribute . . . 756
15.152.2.2 getdata . . . 757
15.152.2.3 getdata1df32 . . . 760
15.152.2.4 getdata1df64 . . . 761
15.152.2.5 getdata1di16 . . . 761
15.152.2.6 getdata1di32 . . . 761
15.152.2.7 getdata1di64 . . . 762
15.152.2.8 getdata1di8 . . . 762
15.152.2.9 getdata2df32 . . . 762

15.152.2.10 getdata2df64 . . . 763
15.152.2.11 getdata2di16 . . . 763
15.152.2.12 getdata2di32 . . . 763
15.152.2.13 getdata2di64 . . . 764
15.152.2.14 getdata2di8 . . . 764
15.152.2.15 getdata3df32 . . . 764
15.152.2.16 getdata3df64 . . . 765
15.152.2.17 getdata3di16 . . . 765
15.152.2.18 getdata3di32 . . . 765
15.152.2.19 getdata3di64 . . . 766
15.152.2.20 getdata3di8 . . . 766
15.152.2.21 getdata4df32 . . . 766
15.152.2.22 getdata4df64 . . . 767
15.152.2.23 getdata4di16 . . . 767
15.152.2.24 getdata4di32 . . . 767
15.152.2.25 getdata4di64 . . . 768
15.152.2.26 getdata4di8 . . . 768
15.152.2.27 getdata5df32 . . . 768
15.152.2.28 getdata5df64 . . . 769
15.152.2.29 getdata5di16 . . . 769
15.152.2.30 getdata5di32 . . . 769
15.152.2.31 getdata5di64 . . . 770
15.152.2.32 getdata5di8 . . . 770
15.152.2.33 getdatascalarf32 . . . 770
15.152.2.34 getdatascalarf64 . . . 771
15.152.2.35 getdatascalari16 . . . 771
15.152.2.36 getdatascalari32 . . . 771
15.152.2.37 getdatascalari64 . . . 772
15.152.2.38 getdatascalari8 . . . 772
15.152.2.39 getdimensions . . . 772
15.152.2.40 getdtype . . . 773
15.152.2.41 getfillvalue . . . 773
15.152.2.42 getname . . . 774
15.152.2.43 getnodimensions . . . 775
15.152.2.44 getshape . . . 775
15.152.2.45 getvariableattributechar . . . 775
15.152.2.46 getvariableattributef32 . . . 776
15.152.2.47 getvariableattributef64 . . . 776
15.152.2.48 getvariableattributei16 . . . 776
15.152.2.49 getvariableattributei32 . . . 777

15.152.2.50 getvariableattributei64 . . . 777
15.152.2.51 getvariableattributei8 . . . 777
15.152.2.52 getvariablefillvaluef32 . . . 778
15.152.2.53 getvariablefillvaluef64 . . . 778
15.152.2.54 getvariablefillvaluei16 . . . 778
15.152.2.55 getvariablefillvaluei32 . . . 779
15.152.2.56 getvariablefillvaluei64 . . . 779
15.152.2.57 getvariablefillvaluei8 . . . 779
15.152.2.58 hasattribute . . . 780
15.152.2.59 initncvariable . . . 780
15.152.2.60 unlimited . . . 780
15.152.2.61 setattribute . . . 781
15.152.2.62 setdata . . . 782
15.152.2.63 setdata1df32 . . . 784
15.152.2.64 setdata1df64 . . . 785
15.152.2.65 setdata1di16 . . . 785
15.152.2.66 setdata1di32 . . . 785
15.152.2.67 setdata1di64 . . . 786
15.152.2.68 setdata1di8 . . . 786
15.152.2.69 setdata2df32 . . . 786
15.152.2.70 setdata2df64 . . . 787
15.152.2.71 setdata2di16 . . . 787
15.152.2.72 setdata2di32 . . . 787
15.152.2.73 setdata2di64 . . . 788
15.152.2.74 setdata2di8 . . . 788
15.152.2.75 setdata3df32 . . . 788
15.152.2.76 setdata3df64 . . . 789
15.152.2.77 setdata3di16 . . . 789
15.152.2.78 setdata3di32 . . . 789
15.152.2.79 setdata3di64 . . . 790
15.152.2.80 setdata3di8 . . . 790
15.152.2.81 setdata4df32 . . . 790
15.152.2.82 setdata4df64 . . . 791
15.152.2.83 setdata4di16 . . . 791
15.152.2.84 setdata4di32 . . . 791
15.152.2.85 setdata4di64 . . . 792
15.152.2.86 setdata4di8 . . . 792
15.152.2.87 setdata5df32 . . . 792
15.152.2.88 setdata5df64 . . . 793
15.152.2.89 setdata5di16 . . . 793

15.152.2.90 setdata5di32 . . . 793
15.152.2.91 setdata5di64 . . . 794
15.152.2.92 setdata5di8 . . . 794
15.152.2.93 setdatascalarf32 . . . 794
15.152.2.94 setdatascalarf64 . . . 795
15.152.2.95 setdatascalari16 . . . 795
15.152.2.96 setdatascalari32 . . . 795
15.152.2.97 setdatascalari64 . . . 796
15.152.2.98 setdatascalari8 . . . 796
15.152.2.99 setfillvalue . . . 796
15.152.2.100 setvariableattributechar . . . 798
15.152.2.101 setvariableattributef32 . . . 798
15.152.2.102 setvariableattributef64 . . . 798
15.152.2.103 setvariableattributei16 . . . 799
15.152.2.104 setvariableattributei32 . . . 799
15.152.2.105 setvariableattributei64 . . . 799
15.152.2.106 setvariableattributei8 . . . 800
15.152.2.107 setvariablefillvaluef32 . . . 800
15.152.2.108 setvariablefillvaluef64 . . . 800
15.152.2.109 setvariablefillvaluei16 . . . 801
15.152.2.110 setvariablefillvaluei32 . . . 801
15.152.2.111 setvariablefillvaluei64 . . . 801
15.152.2.112 setvariablefillvaluei8 . . . 802
15.152.3 Member Data Documentation . . . 802
15.152.3.1 dimension . . . 802
15.152.3.2 dimension . . . 802
15.152.3.3 id . . . 802
15.152.3.4 netcdf . . . 802
15.152.3.5 parent . . . 802
15.152.3.6 s . . . 802
15.152.3.7 the . . . 802
15.152.3.8 the . . . 802
15.153 mo_netcdf::ncvariable Interface Reference . . . 803
15.153.1 Detailed Description . . . 803
15.154 mo_utils::ne Interface Reference . . . 803
15.154.1 Detailed Description . . . 803
15.154.2 Member Function/Subroutine Documentation . . . 803
15.154.2.1 notequal_dp . . . 803
15.154.2.2 notequal_sp . . . 803
15.155 mo_orderpack::nearless Interface Reference . . . 803

15.155.1 Detailed Description . . . 803
15.155.2 Member Function/Subroutine Documentation . . . 804
15.155.2.1 d_nearless . . . 804
15.155.2.2 i_nearless . . . 804
15.155.2.3 r_nearless . . . 804
15.156 mo_spatialsimilarity::nndv Interface Reference . . . 804
15.156.1 Detailed Description . . . 804
15.156.2 Member Function/Subroutine Documentation . . . 805
15.156.2.1 nndv_dp . . . 805
15.156.2.2 nndv_sp . . . 806
15.157 mo_utils::notequal Interface Reference . . . 806
15.157.1 Detailed Description . . . 806
15.157.2 Member Function/Subroutine Documentation . . . 806
15.157.2.1 notequal_dp . . . 807
15.157.2.2 notequal_sp . . . 807
15.158 mo_errormeasures::nse Interface Reference . . . 807
15.158.1 Detailed Description . . . 808
15.158.2 Member Function/Subroutine Documentation . . . 808
15.158.2.1 nse_dp_1d . . . 808
15.158.2.2 nse_dp_2d . . . 808
15.158.2.3 nse_dp_3d . . . 808
15.158.2.4 nse_sp_1d . . . 809
15.158.2.5 nse_sp_2d . . . 809
15.158.2.6 nse_sp_3d . . . 810
15.159 mo_string_utils::num2str Interface Reference . . . 810
15.159.1 Detailed Description . . . 810
15.159.2 Member Function/Subroutine Documentation . . . 811
15.159.2.1 dp2str . . . 811
15.159.2.2 i42str . . . 811
15.159.2.3 i82str . . . 812
15.159.2.4 log2str . . . 812
15.159.2.5 sp2str . . . 813
15.160 mo_string_utils::numarray2str Interface Reference . . . 813
15.160.1 Detailed Description . . . 813
15.160.2 Member Function/Subroutine Documentation . . . 814
15.160.2.1 i4array2str . . . 814
15.161 mo_optimization_utils::objective_interface Interface Reference . . . 814
15.161.1 Detailed Description . . . 814
15.161.2 Constructor & Destructor Documentation . . . 814
15.161.2.1 objective_interface . . . 814

15.162 mo_orderpack::omedian Interface Reference . . . 814
15.162.1 Detailed Description . . . 814
15.162.2 Member Function/Subroutine Documentation . . . 814
15.162.2.1 d_median . . . 814
15.162.2.2 i_median . . . 815
15.162.2.3 r_median . . . 815
15.163 mo_write_fluxes_states::outputdataset Interface Reference . . . 815
15.163.1 Detailed Description . . . 816
15.163.2 Member Function/Subroutine Documentation . . . 816
15.163.2.1 close . . . 816
15.163.2.2 updatedataset . . . 816
15.163.2.3 writetimestep . . . 817
15.163.3 Member Data Documentation . . . 817
15.163.3.1 all . . . 817
15.163.3.2 basin . . . 817
15.163.3.3 count . . . 817
15.163.3.4 counter . . . 817
15.163.3.5 created . . . 818
15.163.3.6 dbasin . . . 818
15.163.3.7 id . . . 818
15.163.3.8 nc . . . 818
15.163.3.9 ncdataset . . . 818
15.163.3.10 steps . . . 818
15.163.3.11 store . . . 818
15.163.3.12 time . . . 818
15.163.3.13 t0 . . . 818
15.163.3.14 variables . . . 818
15.163.3.15 vars . . . 818
15.163.3.16 write . . . 818
15.163.3.17 written . . . 819
15.164 mo_mrm_write_fluxes_states::outputdataset Interface Reference . . . 819
15.164.1 Detailed Description . . . 820
15.164.2 Member Function/Subroutine Documentation . . . 820
15.164.2.1 close . . . 820
15.164.2.2 updatedataset . . . 820
15.164.2.3 writetimestep . . . 821
15.164.3 Member Data Documentation . . . 821
15.164.3.1 all . . . 821
15.164.3.2 basin . . . 821
15.164.3.3 count . . . 821

15.164.3.4 counter . . . 821
15.164.3.5 created . . . 822
15.164.3.6 dbasin . . . 822
15.164.3.7 id . . . 822
15.164.3.8 nc . . . 822
15.164.3.9 ncdataset . . . 822
15.164.3.10 steps . . . 822
15.164.3.11 store . . . 822
15.164.3.12 time . . . 822
15.164.3.13 to . . . 822
15.164.3.14 variables . . . 822
15.164.3.15 vars . . . 822
15.164.3.16 write . . . 822
15.164.3.17 written . . . 823
15.165 no_write_fluxes_states::outputvariable Interface Reference . . . 823
15.165.1 Detailed Description . . . 824
15.165.2 Member Function/Subroutine Documentation . . . 824
15.165.2.1 updatevariable . . . 824
15.165.2.2 writevariabletimestep . . . 825
15.165.3 Member Data Documentation . . . 825
15.165.3.1 average . . . 825
15.165.3.2 avg . . . 826
15.165.3.3 before . . . 826
15.165.3.4 between . . . 826
15.165.3.5 calls . . . 826
15.165.3.6 contains . . . 826
15.165.3.7 count . . . 826
15.165.3.8 counter . . . 826
15.165.3.9 data . . . 826
15.165.3.10 data . . . 826
15.165.3.11 mask . . . 826
15.165.3.12 nc . . . 826
15.165.3.13 ncdataset . . . 826
15.165.3.14 number . . . 827
15.165.3.15 of . . . 827
15.165.3.16 reconstruct . . . 827
15.165.3.17 store . . . 827
15.165.3.18 time . . . 827
15.165.3.19 time . . . 827
15.165.3.20 time . . . 827

15.165.3.21 to . . . 827
15.165.3.22 updatevariable . . . 827
15.165.3.23 variable . . . 827
15.165.3.24 which . . . 827
15.165.3.25 writes . . . 827
15.165.3.26 writing . . . 828
15.166 mo_mrm_write_fluxes_states::outputvariable Interface Reference . . . 828
15.166.1 Detailed Description . . . 829
15.166.2 Member Function/Subroutine Documentation . . . 829
15.166.2.1 updatevariable . . . 829
15.166.2.2 writevariabletimestep . . . 829
15.166.3 Member Data Documentation . . . 830
15.166.3.1 average . . . 830
15.166.3.2 avg . . . 830
15.166.3.3 before . . . 830
15.166.3.4 between . . . 830
15.166.3.5 calls . . . 830
15.166.3.6 contains . . . 830
15.166.3.7 count . . . 830
15.166.3.8 counter . . . 830
15.166.3.9 data . . . 830
15.166.3.10 data . . . 830
15.166.3.11 mask . . . 830
15.166.3.12 rc . . . 830
15.166.3.13 rcdataset . . . 831
15.166.3.14 number . . . 831
15.166.3.15 of . . . 831
15.166.3.16 reconstruct . . . 831
15.166.3.17 store . . . 831
15.166.3.18 time . . . 831
15.166.3.19 time . . . 831
15.166.3.20 time . . . 831
15.166.3.21 to . . . 831
15.166.3.22 updatevariable . . . 831
15.166.3.23 variable . . . 831
15.166.3.24 which . . . 831
15.166.3.25 writes . . . 832
15.166.3.26 writing . . . 832
15.167 mo_append::paste Interface Reference . . . 832
15.167.1 Detailed Description . . . 832

15.167.2 Member Function/Subroutine Documentation . . . 833
15.167.2.1 paste_char_m_m . . . 833
15.167.2.2 paste_char_m_s . . . 833
15.167.2.3 paste_char_m_v . . . 833
15.167.2.4 paste_dp_m_m . . . 833
15.167.2.5 paste_dp_m_s . . . 833
15.167.2.6 paste_dp_m_v . . . 833
15.167.2.7 paste_i4_m_m . . . 833
15.167.2.8 paste_i4_m_s . . . 834
15.167.2.9 paste_i4_m_v . . . 834
15.167.2.10 paste_i8_m_m . . . 834
15.167.2.11 paste_i8_m_s . . . 834
15.167.2.12 paste_i8_m_v . . . 834
15.167.2.13 paste_lgt_m_m . . . 834
15.167.2.14 paste_lgt_m_s . . . 834
15.167.2.15 paste_lgt_m_v . . . 834
15.167.2.16 paste_sp_m_m . . . 834
15.167.2.17 paste_sp_m_s . . . 834
15.167.2.18 paste_sp_m_v . . . 835
15.168 mo_spatialsimilarity::pd Interface Reference . . . 835
15.168.1 Detailed Description . . . 835
15.168.2 Member Function/Subroutine Documentation . . . 836
15.168.2.1 pd_dp . . . 836
15.168.2.2 pd_sp . . . 837
15.169 mo_percentile::percentile Interface Reference . . . 837
15.169.1 Detailed Description . . . 837
15.169.2 Member Function/Subroutine Documentation . . . 837
15.169.2.1 percentile_0d_dp . . . 837
15.169.2.2 percentile_0d_sp . . . 838
15.169.2.3 percentile_1d_dp . . . 838
15.169.2.4 percentile_1d_sp . . . 839
15.170 mo_common_variables::period Type Reference . . . 839
15.170.1 Detailed Description . . . 840
15.170.2 Member Data Documentation . . . 840
15.170.2.1 dend . . . 840
15.170.2.2 dstart . . . 840
15.170.2.3 julend . . . 840
15.170.2.4 julstart . . . 840
15.170.2.5 mend . . . 840
15.170.2.6 mstart . . . 840

15.170.2.7 nobs . . . 840
15.170.2.8 yend . . . 840
15.170.2.9 ystart . . . 841
15.171 mo_percentile::qmedian Interface Reference . . . 841
15.171.1 Detailed Description . . . 841
15.171.2 Member Function/Subroutine Documentation . . . 841
15.171.2.1 qmedian_dp . . . 841
15.171.2.2 qmedian_sp . . . 841
15.172 mo_orderpack::rapknr Interface Reference . . . 842
15.172.1 Detailed Description . . . 842
15.172.2 Member Function/Subroutine Documentation . . . 842
15.172.2.1 d_rapknr . . . 842
15.172.2.2 i_rapknr . . . 842
15.172.2.3 r_rapknr . . . 842
15.173 mo_read_spatial_data::read_spatial_data_ascii Interface Reference . . . 843
15.173.1 Detailed Description . . . 843
15.173.2 Member Function/Subroutine Documentation . . . 843
15.173.2.1 read_spatial_data_ascii_dp . . . 843
15.173.2.2 read_spatial_data_ascii_i4 . . . 844
15.174 mo_corr::realft Interface Reference . . . 845
15.174.1 Detailed Description . . . 845
15.174.2 Member Function/Subroutine Documentation . . . 845
15.174.2.1 realft_dp . . . 845
15.174.2.2 realft_sp . . . 845
15.175 mo_orderpack::refpar Interface Reference . . . 845
15.175.1 Detailed Description . . . 846
15.175.2 Member Function/Subroutine Documentation . . . 846
15.175.2.1 d_refpar . . . 846
15.175.2.2 i_refpar . . . 846
15.175.2.3 r_refpar . . . 846
15.176 mo_orderpack::refsor Interface Reference . . . 846
15.176.1 Detailed Description . . . 846
15.176.2 Member Function/Subroutine Documentation . . . 846
15.176.2.1 d_refsor . . . 846
15.176.2.2 i_refsor . . . 846
15.176.2.3 r_refsor . . . 847
15.177 mo_orderpack::rinpar Interface Reference . . . 847
15.177.1 Detailed Description . . . 847
15.177.2 Member Function/Subroutine Documentation . . . 847
15.177.2.1 d_rinpar . . . 847

15.177.2.2 i_rinpar . . . 847
15.177.2.3 r_rinpar . . . 847
15.178 mo_errormeasures::rmse Interface Reference . . . 847
15.178.1 Detailed Description . . . 848
15.178.2 Member Function/Subroutine Documentation . . . 848
15.178.2.1 rmse_dp_1d . . . 848
15.178.2.2 rmse_dp_2d . . . 848
15.178.2.3 rmse_dp_3d . . . 848
15.178.2.4 rmse_sp_1d . . . 849
15.178.2.5 rmse_sp_2d . . . 849
15.178.2.6 rmse_sp_3d . . . 850
15.179 mo_orderpack::rnkpar Interface Reference . . . 850
15.179.1 Detailed Description . . . 850
15.179.2 Member Function/Subroutine Documentation . . . 850
15.179.2.1 d_rnkpar . . . 850
15.179.2.2 i_rnkpar . . . 850
15.179.2.3 r_rnkpar . . . 850
15.180 mo_errormeasures::sae Interface Reference . . . 851
15.180.1 Detailed Description . . . 851
15.180.2 Member Function/Subroutine Documentation . . . 851
15.180.2.1 sae_dp_1d . . . 851
15.180.2.2 sae_dp_2d . . . 852
15.180.2.3 sae_dp_3d . . . 852
15.180.2.4 sae_sp_1d . . . 852
15.180.2.5 sae_sp_2d . . . 853
15.180.2.6 sae_sp_3d . . . 853
15.181 mo_julian::setcalendar Interface Reference . . . 854
15.181.1 Detailed Description . . . 854
15.181.2 Member Function/Subroutine Documentation . . . 854
15.181.2.1 setcalendarinteger . . . 854
15.181.2.2 setcalendarstring . . . 855
15.182 mo_moment::skewness Interface Reference . . . 855
15.182.1 Detailed Description . . . 855
15.182.2 Member Function/Subroutine Documentation . . . 856
15.182.2.1 skewness_dp . . . 856
15.182.2.2 skewness_sp . . . 856
15.183 mo_orderpack::sort Interface Reference . . . 856
15.183.1 Detailed Description . . . 857
15.183.2 Member Function/Subroutine Documentation . . . 859
15.183.2.1 d_refsor . . . 859

15.183.2.2 l_refsor . . . 859
15.183.2.3 r_refsor . . . 859
15.184 mo_orderpack::sort_index Interface Reference . . . 859
15.184.1 Detailed Description . . . 859
15.184.2 Member Function/Subroutine Documentation . . . 860
15.184.2.1 sort_index_dp . . . 860
15.184.2.2 sort_index_i4 . . . 860
15.184.2.3 sort_index_sp . . . 860
15.185 mo_spatial_agg_disagg_forcing::spatial_aggregation Interface Reference . . . 861
15.185.1 Detailed Description . . . 861
15.185.2 Member Function/Subroutine Documentation . . . 861
15.185.2.1 spatial_aggregation_3d . . . 861
15.185.2.2 spatial_aggregation_4d . . . 862
15.186 mo_spatial_agg_disagg_forcing::spatial_disaggregation Interface Reference . . . 862
15.186.1 Detailed Description . . . 862
15.186.2 Member Function/Subroutine Documentation . . . 863
15.186.2.1 spatial_disaggregation_3d . . . 863
15.186.2.2 spatial_disaggregation_4d . . . 863
15.187 mo_utils::special_value Interface Reference . . . 863
15.187.1 Detailed Description . . . 863
15.187.2 Member Function/Subroutine Documentation . . . 864
15.187.2.1 special_value_dp . . . 864
15.187.2.2 special_value_sp . . . 864
15.188 mo_kind::sprs2_dp Type Reference . . . 865
15.188.1 Detailed Description . . . 865
15.188.2 Member Data Documentation . . . 866
15.188.2.1 irow . . . 866
15.188.2.2 jcol . . . 866
15.188.2.3 len . . . 866
15.188.2.4 n . . . 866
15.188.2.5 val . . . 866
15.189 mo_kind::sprs2_sp Type Reference . . . 866
15.189.1 Detailed Description . . . 867
15.189.2 Member Data Documentation . . . 867
15.189.2.1 irow . . . 867
15.189.2.2 jcol . . . 867
15.189.2.3 len . . . 867
15.189.2.4 n . . . 867
15.189.2.5 val . . . 867
15.190 mo_errormeasures::sse Interface Reference . . . 867

15.190.1 Detailed Description . . . 868
15.190.2 Member Function/Subroutine Documentation . . . 868
15.190.2.1 sse_dp_1d . . . 868
15.190.2.2 sse_dp_2d . . . 868
15.190.2.3 sse_dp_3d . . . 869
15.190.2.4 sse_sp_1d . . . 869
15.190.2.5 sse_sp_2d . . . 870
15.190.2.6 sse_sp_3d . . . 870
15.191 mo_standard_score::standard_score Interface Reference . . . 871
15.191.1 Detailed Description . . . 871
15.191.2 Member Function/Subroutine Documentation . . . 872
15.191.2.1 standard_score_dp . . . 872
15.191.2.2 standard_score_sp . . . 872
15.192 mo_moment::stddev Interface Reference . . . 873
15.192.1 Detailed Description . . . 873
15.192.2 Member Function/Subroutine Documentation . . . 873
15.192.2.1 stddev_dp . . . 873
15.192.2.2 stddev_sp . . . 874
15.193 mo_corr::swap Interface Reference . . . 874
15.193.1 Detailed Description . . . 874
15.194 mo_utils::swap Interface Reference . . . 874
15.194.1 Detailed Description . . . 874
15.194.2 Member Function/Subroutine Documentation . . . 875
15.194.2.1 swap_vec_dp . . . 875
15.194.2.2 swap_vec_i4 . . . 875
15.194.2.3 swap_vec_sp . . . 875
15.194.2.4 swap_xy_dp . . . 875
15.194.2.5 swap_xy_i4 . . . 875
15.194.2.6 swap_xy_sp . . . 875
15.195 mo_global_variables::twsstructure Type Reference . . . 876
15.195.1 Detailed Description . . . 876
15.195.2 Member Data Documentation . . . 876
15.195.2.1 basinid . . . 876
15.195.2.2 fname . . . 876
15.195.2.3 tws . . . 876
15.196 mo_orderpack::uniinv Interface Reference . . . 877
15.196.1 Detailed Description . . . 877
15.196.2 Member Function/Subroutine Documentation . . . 877
15.196.2.1 d_uniinv . . . 877
15.196.2.2 i_uniinv . . . 877

15.196.2.3 r_uniinv . . . 877
15.197 mo_orderpack::unipar Interface Reference . . . 877
15.197.1 Detailed Description . . . 877
15.197.2 Member Function/Subroutine Documentation . . . 877
15.197.2.1 d_unipar . . . 877
15.197.2.2 i_unipar . . . 878
15.197.2.3 r_unipar . . . 878
15.198 mo_orderpack::unirnk Interface Reference . . . 878
15.198.1 Detailed Description . . . 878
15.198.2 Member Function/Subroutine Documentation . . . 878
15.198.2.1 d_unirnk . . . 878
15.198.2.2 i_unirnk . . . 878
15.198.2.3 r_unirnk . . . 878
15.199 mo_orderpack::unista Interface Reference . . . 878
15.199.1 Detailed Description . . . 879
15.199.2 Member Function/Subroutine Documentation . . . 879
15.199.2.1 d_unista . . . 879
15.199.2.2 i_unista . . . 879
15.199.2.3 r_unista . . . 879
15.200 mo_restart::unpack_field_and_write Interface Reference . . . 879
15.200.1 Detailed Description . . . 879
15.200.2 Member Function/Subroutine Documentation . . . 880
15.200.2.1 unpack_field_and_write_1d_dp . . . 880
15.200.2.2 unpack_field_and_write_1d_i4 . . . 880
15.200.2.3 unpack_field_and_write_2d_dp . . . 880
15.200.2.4 unpack_field_and_write_3d_dp . . . 881
15.201 mo_orderpack::valmed Interface Reference . . . 881
15.201.1 Detailed Description . . . 881
15.201.2 Member Function/Subroutine Documentation . . . 881
15.201.2.1 d_valmed . . . 881
15.201.2.2 i_valmed . . . 881
15.201.2.3 r_valmed . . . 882
15.202 mo_orderpack::valnth Interface Reference . . . 882
15.202.1 Detailed Description . . . 882
15.202.2 Member Function/Subroutine Documentation . . . 882
15.202.2.1 d_valnth . . . 883
15.202.2.2 i_valnth . . . 883
15.202.2.3 r_valnth . . . 883
15.203 mo_ncwrite::var2nc Interface Reference . . . 883
15.203.1 Detailed Description . . . 884

15.203.2 Member Function/Subroutine Documentation . . . 885
15.203.2.1 var2nc_1d_dp . . . 885
15.203.2.2 var2nc_1d_i4 . . . 885
15.203.2.3 var2nc_1d_sp . . . 885
15.203.2.4 var2nc_2d_dp . . . 885
15.203.2.5 var2nc_2d_i4 . . . 885
15.203.2.6 var2nc_2d_sp . . . 886
15.203.2.7 var2nc_3d_dp . . . 886
15.203.2.8 var2nc_3d_i4 . . . 886
15.203.2.9 var2nc_3d_sp . . . 886
15.203.2.10 var2nc_4d_dp . . . 886
15.203.2.11 var2nc_4d_i4 . . . 886
15.203.2.12 var2nc_4d_sp . . . 887
15.203.2.13 var2nc_5d_dp . . . 887
15.203.2.14 var2nc_5d_i4 . . . 887
15.203.2.15 var2nc_5d_sp . . . 887
15.204 mo_ncwrite::variable Type Reference . . . 888
15.204.1 Detailed Description . . . 889
15.204.2 Member Data Documentation . . . 889
15.204.2.1 att . . . 889
15.204.2.2 count . . . 889
15.204.2.3 dimids . . . 890
15.204.2.4 dimtypes . . . 890
15.204.2.5 g0_b . . . 890
15.204.2.6 g0_d . . . 890
15.204.2.7 g0_f . . . 890
15.204.2.8 g0_i . . . 890
15.204.2.9 g1_b . . . 890
15.204.2.10 g1_d . . . 890
15.204.2.11 g1_f . . . 890
15.204.2.12 g1_i . . . 890
15.204.2.13 g2_b . . . 890
15.204.2.14 g2_d . . . 890
15.204.2.15 g2_f . . . 891
15.204.2.16 g2_i . . . 891
15.204.2.17 g3_b . . . 891
15.204.2.18 g3_d . . . 891
15.204.2.19 g3_f . . . 891
15.204.2.20 g3_i . . . 891
15.204.2.21 g4_b . . . 891

15.204.2.23 g4_d . . . 891
15.204.2.24 g4_f . . . 891
15.204.2.25 g4_i . . . 891
15.204.2.26 name . . . 891
15.204.2.27 natt . . . 891
15.204.2.28 ndims . . . 892
15.204.2.29 nvls . . . 892
15.204.2.30 nsubs . . . 892
15.204.2.31 start . . . 892
15.204.2.32 unlimited . . . 892
15.204.2.33 varid . . . 892
15.204.2.34 varflag . . . 892
15.204.2.35 vtype . . . 892
15.205 mo_moment::variance Interface Reference . . . 892
15.205.1 Detailed Description . . . 892
15.205.2 Member Function/Subroutine Documentation . . . 893
15.205.2.1 variance_dp . . . 893
15.205.2.2 variance_sp . . . 893
15.206 mo_errormeasures::wnse Interface Reference . . . 893
15.206.1 Detailed Description . . . 894
15.206.2 Member Function/Subroutine Documentation . . . 894
15.206.2.1 wnse_dp_1d . . . 894
15.206.2.2 wnse_dp_2d . . . 894
15.206.2.3 wnse_dp_3d . . . 894
15.206.2.4 wnse_sp_1d . . . 895
15.206.2.5 wnse_sp_2d . . . 895
15.206.2.6 wnse_sp_3d . . . 896
15.207 mo_xor4096::xor4096 Interface Reference . . . 896
15.207.1 Detailed Description . . . 896
15.207.2 Member Function/Subroutine Documentation . . . 896
15.207.2.1 xor4096d_0d . . . 896
15.207.2.2 xor4096d_1d . . . 896
15.207.2.3 xor4096f_0d . . . 897
15.207.2.4 xor4096f_1d . . . 897
15.207.2.5 xor4096l_0d . . . 897
15.207.2.6 xor4096l_1d . . . 897
15.207.2.7 xor4096s_0d . . . 897
15.207.2.8 xor4096s_1d . . . 897
15.208 mo_xor4096::xor4096g Interface Reference . . . 897
15.208.1 Detailed Description . . . 897

15.208.2 Member Function/Subroutine Documentation . . . 898
15.208.2.1 xor4096gd_0d . . . 898
15.208.2.2 xor4096gd_1d . . . 898
15.208.2.3 xor4096gf_0d . . . 898
15.208.2.4 xor4096gf_1d . . . 898

16 File Documentation **899**
16.1 1-main.dox File Reference . . . 899
16.2 2-get_started.dox File Reference . . . 899
16.3 3-data_preparation.dox File Reference . . . 899
16.4 4-visualise_out.dox File Reference . . . 899
16.5 5-calibration.dox File Reference . . . 899
16.6 6-style_guide.dox File Reference . . . 899
16.7 7-test_basin.dox File Reference . . . 899
16.8 8-protocols_for_setup_new_mHM_basin.dox File Reference . . . 899
16.9 9-mRM.md File Reference . . . 899
16.10 DEPENDENCIES.md File Reference . . . 899
16.11 mhm_driver.f90 File Reference . . . 899
16.11.1 Function/Subroutine Documentation . . . 899
16.11.1.1 mhm_driver . . . 900
16.12 mhm_papers.md File Reference . . . 902
16.13 mo_anneal.f90 File Reference . . . 902
16.14 mo_append.f90 File Reference . . . 902
16.15 mo_canopy_interc.f90 File Reference . . . 902
16.16 mo_common_constants.f90 File Reference . . . 902
16.17 mo_common_file.f90 File Reference . . . 902
16.18 mo_common_functions.f90 File Reference . . . 903
16.19 mo_common_read_config.f90 File Reference . . . 903
16.20 mo_common_read_data.f90 File Reference . . . 903
16.21 mo_common_restart.f90 File Reference . . . 903
16.22 mo_common_variables.f90 File Reference . . . 903
16.23 mo_constants.f90 File Reference . . . 903
16.24 mo_corr.f90 File Reference . . . 904
16.25 mo_dds.f90 File Reference . . . 904
16.26 mo_errormeasures.f90 File Reference . . . 904
16.27 mo_file.f90 File Reference . . . 904
16.28 mo_finish.f90 File Reference . . . 905
16.29 mo_global_variables.f90 File Reference . . . 905
16.30 mo_grid.f90 File Reference . . . 905
16.31 mo_init_states.f90 File Reference . . . 905

16.32 mo_julian.f90 File Reference . . . 905
16.33 mo_kind.f90 File Reference . . . 905
16.34 mo_linfit.f90 File Reference . . . 906
16.35 mo_mad.f90 File Reference . . . 906
16.36 mo_mcmc.f90 File Reference . . . 906
16.37 mo_message.f90 File Reference . . . 906
16.38 mo_meteo_forcings.f90 File Reference . . . 906
16.39 mo_mhm.f90 File Reference . . . 906
16.40 mo_mhm_constants.f90 File Reference . . . 907
16.41 mo_mhm_eval.f90 File Reference . . . 907
16.42 mo_mhm_read_config.f90 File Reference . . . 907
16.43 mo_moment.f90 File Reference . . . 907
16.44 mo_mrm_constants.f90 File Reference . . . 907
16.45 mo_mrm_eval.f90 File Reference . . . 908
16.46 mo_mrm_file.f90 File Reference . . . 908
16.47 mo_mrm_global_variables.f90 File Reference . . . 908
16.48 mo_mrm_init.f90 File Reference . . . 908
16.49 mo_mrm_mpr.f90 File Reference . . . 908
16.50 mo_mrm_net_startup.f90 File Reference . . . 908
16.50.1 Function/Subroutine Documentation . . . 909
16.50.1.1 calculate_l11_flow_accumulation . . . 909
16.51 mo_mrm_objective_function_runoff.f90 File Reference . . . 909
16.52 mo_mrm_read_config.f90 File Reference . . . 909
16.53 mo_mrm_read_data.f90 File Reference . . . 909
16.54 mo_mrm_restart.f90 File Reference . . . 909
16.55 mo_mrm_river_head.f90 File Reference . . . 910
16.56 mo_mrm_routing.f90 File Reference . . . 910
16.57 mo_mrm_signatures.f90 File Reference . . . 910
16.58 mo_mrm_write.f90 File Reference . . . 910
16.59 mo_mrm_write_fluxes_states.f90 File Reference . . . 910
16.60 mo_ncread.f90 File Reference . . . 910
16.61 mo_ncwrite.f90 File Reference . . . 911
16.62 mo_netcdf.f90 File Reference . . . 911
16.63 mo_neutrons.f90 File Reference . . . 911
16.64 mo_nml.f90 File Reference . . . 911
16.65 mo_objective_function.f90 File Reference . . . 911
16.66 mo_optimization_utils.f90 File Reference . . . 912
16.67 mo_orderpack.f90 File Reference . . . 912
16.68 mo_percentile.f90 File Reference . . . 912
16.69 mo_pet.f90 File Reference . . . 913

16.70 mo_read_forcing_nc.f90 File Reference . . . 913
16.71 mo_read_latlon.f90 File Reference . . . 913
16.72 mo_read_optional_data.f90 File Reference . . . 913
16.73 mo_read_spatial_data.f90 File Reference . . . 913
16.74 mo_read_timeseries.f90 File Reference . . . 913
16.75 mo_restart.f90 File Reference . . . 914
16.76 mo_runoff.f90 File Reference . . . 914
16.77 mo_sce.f90 File Reference . . . 914
16.77.1 Function/Subroutine Documentation . . . 914
16.77.1.1 set_optional . . . 914
16.77.1.2 write_best_final . . . 915
16.77.1.3 write_best_intermediate . . . 915
16.77.1.4 write_population . . . 915
16.77.1.5 write_termination_case . . . 916
16.78 mo_set_netcdf_outputs.f90 File Reference . . . 916
16.79 mo_snow_accum_melt.f90 File Reference . . . 916
16.80 mo_soil_moisture.f90 File Reference . . . 916
16.81 mo_spatial_agg_disagg_forcing.f90 File Reference . . . 916
16.82 mo_spatialsimilarity.f90 File Reference . . . 917
16.83 mo_standard_score.f90 File Reference . . . 917
16.84 mo_startup.f90 File Reference . . . 917
16.85 mo_string_utils.f90 File Reference . . . 917
16.86 mo_template.f90 File Reference . . . 917
16.87 mo_temporal_aggregation.f90 File Reference . . . 918
16.88 mo_temporal_disagg_forcing.f90 File Reference . . . 918
16.89 mo_timer.f90 File Reference . . . 918
16.90 mo_utils.f90 File Reference . . . 918
16.91 mo_write_ascii.f90 File Reference . . . 919
16.92 mo_write_fluxes_states.f90 File Reference . . . 919
16.93 mo_xor4096.f90 File Reference . . . 919
16.94 mrm_driver.f90 File Reference . . . 919
16.94.1 Function/Subroutine Documentation . . . 919
16.94.1.1 mrm_driver . . . 919
16.95 RELEASES.md File Reference . . . 921

Chapter 1

Introduction to mHM

This chapter is divided in the following sections:

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

1.1 Short Description

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

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

1.2 The Grid-based mHM Model

The mHM hydrologic model is based on numerical approximations of dominant hydrological processes that have been tested in various models: HBV [**?**], [**?**], [**?**] and VIC [**?**] . 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 [**?**] .

Figure 1.1: Typical mHM cell

Dominant processes of the hydrological cycle at mesoscale span over several orders of magnitude [**?**] . 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 [**?**] 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, $\mathrm{mm\ d^{-1}}$

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* [**?**] of feasible solutions. Moreover, the high dimensionality of this optimization problem is also a daunting task for the state-of-the-art optimization algorithms [**?**] . To overcome this problem a

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

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:

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

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 [**?**], dynamically dimensioned search [**?**] . 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) [**?**] . 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 [**?**] . 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 [**?**] and w_i denote the degree of importance of each objective. High values of p, say $p=6$, should be chosen to avoid substitution of objective function values at low levels. In general, the estimators related to daily streamflows were twice as important as the long-term ones, thus $\{w_i\} = \{\frac{2}{4},\frac{1}{4},\frac{1}{4},\frac{2}{4}\}$. The NSE for a given time interval t' is given by

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

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

1.7 Helpful links

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

1.8 Test basin

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

1.9 Protocols

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

Chapter 2

Getting Started

This chapter will introduce the structure of mHM technically.

2.1 Receive mHM

As from June 2018, the current release of mHM is available through a code repository provided by the version control system GitLab (https://git.ufz.de/mhm/mhm). Access GitLab for external users can be granted upon request through email to mhm-admin@ufz.de. Additionally, a mirror of the current release is available on GitHub without registration needed (https://github.com/mhm-ufz/mhm). Note that our repository on GibHub.com will be always a mirror of the current release at GitLab.

2.2 Install NETCDF

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

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

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

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

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

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

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

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

```
cd /Downloads/zlib-1.2.11
```

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

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

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

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

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

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

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

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

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

2.2.2 Short Guide Install to NETCDF on MacOS

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

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

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

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

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

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

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

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

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

For NAG compiler only:

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

For gfortran compiler only:

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

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

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

2.3 Run mHM under CYGWIN on Windows

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

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

Figure 2.1: netcdf packages for CYGWIN

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

Figure 2.2: hdf5 packages for CYGWIN

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

Figure 2.3: gfortran packages for CYGWIN

Figure 2.4: libgfortran packages for CYGWIN

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

Figure 2.5: make packages for CYGWIN

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

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

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

Figure 2.6: setup environmental path CYGWIN

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

2.4 Compile mHM

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

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

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

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

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

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

2.5 Test mHM on Example Basin

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

```
test_basin/
```

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

```
./mhm
```

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

2.6 Run your own Simulation

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

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

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

2.6.1 Main Configuration: Paths, Periods, Switches

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

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

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

2.6.2 Output Configuration: Time Steps, States, Fluxes

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

2.6.3 Regionalised Parameters: Initial Values and Ranges

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

2.7 Calibration and Optimization

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

2.7.1 The Optimization Routines

mHM comes with four available optimization methods:

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

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

Objective functions currently implemented in mHM are:

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

2.7.2 Calibration Settings for mHM

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

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

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

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

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

2.7.3 Final Calibration Results

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

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

You can run mHM directly with the generated namelist (by renaming it) or incorporate the results in `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-WFD↩EI.pdf).

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

The two meteorological variables which are needed:

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

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

- processCase(5) = 0 - PET is read from input.
- processCase(5) = 1 - Hargreaves-Samani equation

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

Name	Unit	Temporal resolution
0 - Potential evapotranspiration	mm	hourly to daily
1 - Minimum air temperature	°C	daily
1 - Maximum air temperature	°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" (∼ 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/CORO-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 Elevation 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 NCVI↩EW using 8-flow direction convention (1,2,4,8,16,32,64,128) (i.e., 1 pointing to the east and then moving clockwise every 45 $^\circ$) have to be rotated accordingly. In other words, direction 1 should be interpreted as 64, 2 as 128, and so on. All 2D variables, however, included the previous one, will produce consistent maps as that shown in (Moselle Basin) if plotted with appropriate routines (e.g., Python or NCL) that take into account the geographic coordinates stored within the NetCDF file.

Chapter 5

Calibration Options

5.1 Calibration of Parameters on Observations

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

5.1.1 Parameters

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

5.1.2 Methods

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

5.1.3 Functions

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

5.1.4 Data

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

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

5.1.5 Output

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

Chapter 6

Coding and Documentation Style

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

General

- **Follow the coding and documentation style of template mo_template.f90 very closely.**
- Enforce SI units in all code, with the only exception of mms^{-1} for $\mathrm{kg\ m}^{-2}\mathrm{s}^{-1}$.
- 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

Short Description

This section provides information on how to run the routing model mRM as a stand-alone program outside of mHM

Compiling mRM

Test basin

Chapter 10

mHM Dependencies

NetCDF4

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

Chapter 11

Publications using mHM

- 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): Cosmicray neutron rover surveys of field soil moisture and the influence of roads, Water Resour. Res., https://doi.org/10.1029/2017WR021719
- Thober, S., Kumar, R., Wanders, N., Marx, A., Pan, M., Rakovec, O., Samaniego, L., Sheffield, J., Wood, E.F., Zink, M. (2018): Multi-model ensemble projections of European river floods and high flows at 1.5, 2, and 3 degrees global warming, Environ. Res. Lett. 13 (1), art. 014003, http://iopscience.iop.↩org/article/10.1088/1748-9326/aa9e35
- Zink, M., Mai, J., Cuntz, M., Samaniego, L. (2018): Conditioning a hydrologic model using patterns of remotely sensed land surface temperature, Water Resour. Res. 54 (4), 2976 - 2998, https://doi.org/10.↩1002/2017WR021346

- Samaniego, L., Kumar, R., Thober, S., Rakovec, O., Zink, M., Wanders, N., Eisner, S., Müller Schmied, H., Sutanudjaja, E.H., Warrach-Sagi, K., Attinger, S., (2017): Toward seamless hydrologic predictions across spatial scales, Hydrol. Earth Syst. Sci. 21 (9), 4323 - 4346, https://www.hydrol-earth-syst-sci.net/21/4323/2017/
- Samaniego, L., Kumar, R., Breuer, L., Chamorro, A., Flörke, M., Pechlivanidis, I.G., Schäfer, D., Shah, H., Vetter, T., Wortmann, M., Zeng, X., (2017): Propagation of forcing and model uncertainties on to hydrological drought characteristics in a multi-model century-long experiment in large river basins, Clim. Change 141 (3), 435 - 449, https://link.springer.com/article/10.1007/s10584-016-1778-y
- Vigiak, O., Lutz, S., Mentzafou, A., Chiogna, G., Ye, T., Majone, B., Beck, H., de Roo, A., Malagó, A., Bouraoui, F., Kumar, R., Samaniego, L., Merz, R., Gamvroudis, C., Skoulikidis, N., Nikolaidis, N.P., Bellin, A., Acuňa, V., Mori, N., Ludwig, R., Pistocchi, A., (2018): Uncertainty of modelled flow regime for flow-ecological assessment in Southern Europe, Sci. Total Environ. 615 , 1028 - 1047, https://www.sciencedirect.com/science/article/pii/S0048969717326475
- Eisner, S., Flörke, M., Chamorro, A., Daggupati, P., Donnelly, C., Huang, J., Hundecha, Y., Koch, H., Kalugin, A., Krylenko, I., Mishra, V., Piniewski, M., Samaniego, L., Seidou, O., Wallner, M., Krysanova, V., (2017): An ensemble analysis of climate change impacts on streamflow seasonality across 11 large river basins, Clim. Change 141 (3), 401 - 417, https://link.springer.com/article/10.1007/s10584-016-1844-5
- Pechlivanidis, I.G., Arheimer, B., Donnelly, C., Hundecha, Y., Huang, S., Aich, V., Samaniego, L., Eisner, S. and Shi, P., (2017): Analysis of hydrological extremes at different hydro-climatic regimes under present and future conditions. Climatic Change, 141(3), pp.467-481, https://link.springer.com/article/10.1007/s10584-016-1723-0
- Hattermann, F.F., Krysanova, V., Gosling, S.N., Dankers, R., Daggupati, P., Donnelly, C., Flörke, M., Huang, S., Motovilov, Y., Buda, S., Yang, T., Müller, C., Leng, G., Tang, Q., Portmann, F.T., Hagemann, S., Gerten, D., Wada, Y., Masaki, Y., Alemayehu, T., Satoh, Y., Samaniego, L., (2017): Crossscale 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:10.1002/2016↩WR019430, http://onlinelibrary.wiley.com/doi/10.1002/2016WR019430/full

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

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

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

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

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

- Woehling, T., Samaniego, L., & Kumar, R. (2013). Evaluating multiple performance criteria to calibrate the distributed hydrological model of the upper Neckar catchment. Environmental Earth Sciences, 69(2), 453–468. doi:10.1007/s12665-013-2306-2, https://link.springer.com/article/10.↩1007/s12665-013-2306-2

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

- Kumar, R., Livneh, B., & Samaniego, L. (2013). Toward computationally efficient large-scale hydrologic predictions with a multiscale regionalization scheme. Water Resources Research, 49(9), 5700–5714. doi:10.↩1002/wrcr.20431, http://onlinelibrary.wiley.com/doi/10.1002/wrcr.20431/full

- Kumar, R., Samaniego, L., & Attinger, S. (2013). Implications of distributed hydrologic model parameterization on water fluxes at multiple scales and locations. Water Resources Research, 49(1), 360–379. doi:10.1029/2012WR012195, http://onlinelibrary.wiley.com/doi/10.1029/2012WR012195/abstract
- Zacharias, S., Bogena, H., Samaniego, L., Mauder, M., Fuß, R., Puetz, T., ... & Bens, O. (2011). A network of terrestrial environmental observatories in Germany. Vadose Zone Journal, 10(3), 955-973, https://dl.sciencesocieties.org/publications/vzj/abstracts/10/3/955
- Kalbacher, T., Delfs, J. O., Shao, H., Wang, W., Walther, M., Samaniego, L., ... & Sun, F. (2012). The IWAS-ToolBox: software coupling for an integrated water resources management. Environmental Earth Sciences, 65(5), 1367-1380, https://link.springer.com/article/10.1007/s12665-011-1270-y
- Samaniego, L., Kumar, R., & Jackisch, C. (2011). Predictions in a data-sparse region using a regionalized grid-based hydrologic model driven by remotely sensed data. Hydrology Research, 42(5), 338–355. doi:10.2166/nh.2011.156, http://hr.iwaponline.com/content/42/5/338.article-info
- Kumar, R., Samaniego, L., & Attinger, S. (2010). The effects of spatial discretization and model parameterization on the prediction of extreme runoff characteristics. Journal of Hydrology, 392(1-2), 54–69. doi:10.1016/j.jhydrol.2010.07.047, http://www.sciencedirect.com/science/article/pii/S0022169410004865
- Samaniego, L., Kumar, R., & Attinger, S. (2010). Multiscale parameter regionalization of a grid-based hydrologic model at the mesoscale. Water Resources Research, 46(5), W05523. doi:10.1029/2008WR007327, http://onlinelibrary.wiley.com/doi/10.1029/2008WR007327/full

Chapter 12

mHM RELEASE NOTES

mHM v5.9 (July 2018)

New Features:

- Major restructuralization of the mHM code.
- MPR is now executed before mHM is run.
- MPR can be compiled as a standalone tool.
- mHM without mRM can be compiled.
- The code in general is strictly reorganized into modules that belong to their respective processes (MPR, mHM, mRM). This is not only done for constants, global variables and so on, but also for every module and the reading of input as well as the namelists.
- This leads to the creation of those folders:
  - `./common` (code shared by MPR, mHM and mRM) included for every compilation option
  - `./lib` (code shared by MPR, mHM and mRM) included for every compilation option
  - `./MPR` (code for MPR), not included for mRM standalone
  - `./common_mHM_mRM` (code shared by mHM and mRM), not included for MPR standalone
  - `./mHM` (code for mHM), not included for MPR and mRM standalone
  - `./mRM` (code for mRM), not included for MPR standalone and mHM without routing
- Code is reformatted (indentation=2, spacing unified)
- Removed many duplicate code parts (e.g. shared between mHM and mRM)
- Check cases are minimized (reduced output, shorter time periods (<=2 years), less basins)
- Check cases can now be set up more easily (by use of model_wrapper for automatic creation of new nml files)
- Check cases now run a python script for output comparison, advantage: tolerance now also allowed for ascii-output and support of 4-D netCDF files without time dimension
- fSealed is now an effective parameter
- mHM effective parameters now all have three dimensions internally (nCells_L1, [iHorizon, LAI-Time], nLCoverScenes)
- Introduction of new derived types:
  - `Grid` (merge of basin_info, basin_info_mrm, gridGeoRef, nCells, longitude, latitude, Id) used for each level (0, 1, 11, 2) individually

  - `GridRemapper` (merge of lower_bound, upper_bound, etc.)

- mhm_eval and mrm_eval now have common procedure interface (needed for fully flexible optimisation)
- A new post-processor can check and adapt some fields for doxygen generation
- Changes that are not backwards-compatible:
  - Restart files are restructured (now contain only the minimum required for restart):
  - MPR: effective parameters at L1 + grid information
  - mHM: effective parameters, states and fluxes at L1 + grid information
  - mRM: routing-specific parameters, states, fluxes, configs at L1/L11 + grid information
  - mhm.nml is restructured and not backwards compatible mainly due to the reorganization of modules in dependency of their processes
  - gridded LAI values are now used for all effective parameters (no fallback to LAIclasses anymore)
  - canopy height used for aerodynamic resistance is now scaled with the actual LAI timeseries and not with a dummy timeseries of intensive orchard
- Considerable improvements and reduced redundancy in estimation of an empirical distribution of slopes (sort function in the mo_startup). Example for the Australian domain (180 million L0 cells), it reduced time for sorting slope from 32hours to only 1 minute.
- mtCLIM preprocessor in pre-proc/mtCLIM, based on `mtCLIM v4.3`. This code is able to estimate humidity (vapore pressure or vapore pressure deficit) and incoming shortwave radiation based on meteorological variables (minimum and maximum air temperature, precipitation) and morphological characteristics of the underlying terrain (digital elevation model, slope, aspect).
- mHM2OGS preprocessor in pre-proc/GIS2FEM3. This code converts the triangular-wise or quadrilateral-wise recharge data from mHM into the nodal source terms of a three dimensional finite element model for `OGS`.
- Updated documentation for CYGWIN installations under Windows 7 and 10.
- Added new objective function number 31: weighted NSE (NSE is weighted with observed discharge)
- Removal of bin files and related code (only nc and ascii files are used)

Bugs resolved from release 5.8:

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

Known bugs:

None.

Restrictions:

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

mHM v5.8 (Dec 2017)

New Features:

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

Bugs resolved from release 5.7:

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

Known bugs:

None.

Restrictions:

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

mHM v5.7 (Jun 2017)

New Features:

- New process descriptions for the soil evapotranspiration module:
  - Field capacity dependency to root fraction coefficient (`processCase(3)=2`) – implemented by *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`.

mHM v5.6 (Dec 2016)

New Features:

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

Bugs resolved:

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

Known bugs:

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

Restrictions:

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

mHM v5.5 (Jun 2016)

New Features:

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

Bugs resolved:

- Assigning routing parameters is done properly now.

Known bugs:

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

Restrictions:

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

mHM v5.4 (Dec 2015)

New Features:

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

Bugs resolved:

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

Known bugs:

- None.

Restrictions:

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

mHM v5.3 (Jun 2015)

New Features:

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

Bugs resolved:

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

Known bugs:

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

Restrictions:

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

mHM 5.2 (Dec 2014)

New Features:

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

Bugs resolved:

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

Known bugs:

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

mHM 5.1 (Jun 2014)

New Features:

- OpenMP handling of routines such as the multi-scale parameter regionalization (MPR)
- Multi-scale implementation, i.e. running mHM simultaneously in several basins with different resolutions
- Automatic check case framework, i.e. testing new implementations on their validity and back-compatibility
- Implementation of inflow gauges, i.e. feeding discharge time series from upstream areas at catchment boundaries
- File gaugeinfo.txt specifying gauging stations is now part of namelist mhm.nml
- Code is now free of Numerical Recipes proprietary code

- Can now run on a single cell (no routing performed) Hydrological modelling resolution (L1) equal to morphological input data resolution (L0) possible
- Windows compatible (with Cygwin)
- Support of regular geographic coordinate systems (e.g lat-lon) in addition to equal-area coordinate systems (UTM)

Bugs resolved:

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

Known bugs:

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

mHM 5.0 (Dec 2013)

New Features:

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

Known bugs:

- None.

Chapter 13

Data Type Index

13.1 Data Types List

Here are the data types with brief descriptions:

mo_moment::absdev . . . 73
mo_anneal::anneal
Anneal . . . 74
mo_append::append
Append (rows) scalars, vectors, and matrixes onto existing array . . . 76
mo_corr::arth . . . 80
mo_ncwrite::attribute . . . 81
mo_corr::autocoeffk . . . 82
mo_corr::autocorr . . . 83
mo_moment::average . . . 85
mo_mrm_global_variables::basininfo_mrm . . . 86
mo_errormeasures::bias . . . 88
mo_moment::central_moment . . . 91
mo_moment::central_moment_var . . . 92
mo_standard_score::classified_standard_score
Calculates the classified standard score (e.g. classes are months) . . . 93
mo_corr::corr . . . 95
mo_moment::correlation . . . 96
mo_moment::covariance . . . 97
mo_corr::crosscoeffk . . . 98
mo_corr::crosscorr . . . 99
mo_orderpack::ctrper . . . 100
mo_temporal_aggregation::day2mon_average
Day-to-month average (day2mon_average) . . . 101
mo_ncwrite::dims . . . 102
dummy_mrm . . . 103
mo_ncwrite::dump_netcdf . . . 103
mo_utils::eq . . . 106
mo_utils::equal
Comparison of real values . . . 106
mo_optimization_utils::eval_interface . . . 108
mo_orderpack::fndnth . . . 108
mo_corr::four1 . . . 109
mo_corr::fourrow . . . 109
mo_mrm_global_variables::gaugingstation . . . 110
mo_utils::ge . . . 111
mo_anneal::generate_neighborhood_weight . . . 111
mo_ncread::get_ncvar . . . 112

mo_xor4096::get_timeseed . . . 116
mo_anneal::gettemperature
GetTemperature . . . 117
mo_utils::greaterequal . . . 118
mo_common_variables::grid . . . 120
mo_common_variables::gridremapper . . . 122
mo_temporal_aggregation::hour2day_average
Hour-to-day average (hour2day_average) . . . 124
mo_orderpack::indmed . . . 125
mo_orderpack::indnth . . . 125
mo_orderpack::inspar . . . 126
mo_orderpack::inssor . . . 127
mo_utils::is_finite
.true. if not IEEE Inf, IEEE NaN, nor IEEE Inf nor IEEE NaN, respectively . . . 128
mo_utils::is_nan . . . 130
mo_utils::is_normal . . . 131
mo_errormeasures::kge
Kling-Gupta-Efficiency measure . . . 132
mo_errormeasures::kgenocorr
Kling-Gupta-Efficiency measure without correlation . . . 135
mo_moment::kurtosis . . . 138
mo_utils::le . . . 139
mo_utils::lesserequal . . . 140
mo_linfit::linfit
Fits a straight line to input data by minimizing chi^2 . . . 141
mo_errormeasures::lnnse . . . 143
mo_utils::locate
Find closest values in a monotonic series, returns the indexes . . . 145
mo_mad::mad . . . 148
mo_errormeasures::mae . . . 150
mo_mcmc::mcmc
This module is Monte Carlo Markov Chain sampling of a posterior parameter distribution (measurement errors are either known or modeled) . . . 152
mo_mcmc::mcmc_stddev
This module is Monte Carlo Markov Chain sampling of a posterior parameter distribution (measurement errors are neither known nor modeled) . . . 156
mo_template::mean
The average . . . 160
mo_moment::mean . . . 162
mo_percentile::median . . . 162
mo_moment::mixed_central_moment . . . 163
mo_moment::mixed_central_moment_var . . . 164
mo_anneal . . . 166
mo_append
Append values on existing arrays . . . 169
mo_canopy_interc
Canopy interception . . . 175
mo_common_constants
Provides constants commonly used by mHM, mRM and MPR . . . 176
mo_common_file
Provides file names and units for mRM . . . 179
mo_common_functions
Provides small utility functions used by multiple parts of the code (mHM, mRM, MPR) . . . 181
mo_common_read_config
Reading of main model configurations . . . 182
mo_common_read_data
TODO: add description . . . 184

mo_common_restart
TODO: add description . . . 186
mo_common_variables
Provides structures needed by mHM, mRM and/or mpr . . . 189
mo_constants
Provides computational, mathematical, physical, and file constants . . . 194
mo_corr . . . 203
mo_dds
Dynamically Dimensioned Search (DDS) . . . 212
mo_errormeasures . . . 216
mo_file
Provides file names and units for mHM . . . 247
mo_finish
Finish a program gracefully . . . 249
mo_global_variables
Global variables ONLY used in reading, writing and startup . . . 251
mo_grid
TODO: add description . . . 262
mo_init_states
Initialization of all state variables of mHM . . . 268
mo_julian
Julian date conversion routines . . . 270
mo_kind
Define number representations . . . 296
mo_linfit
Fitting a straight line . . . 299
mo_mad . . . 300
mo_mcmc
This module is Monte Carlo Markov Chain sampling of a posterior parameter distribution . . . 302
mo_message
Write out concatenated strings . . . 305
mo_meteo_forcings
Prepare meteorological forcings data for mHM . . . 307
mo_mhm
Call all main processes of mHM . . . 314
mo_mhm_constants
Provides mHM specific constants . . . 322
mo_mhm_eval
Runs mhm with a specific parameter set and returns required variables, e.g. runoff . . . 326
mo_mhm_read_config
Reading of main model configurations . . . 330
mo_moment . . . 332
mo_mrm_constants
Provides mRM specific constants . . . 344
mo_mrm_eval
Runs mrm with a specific parameter set and returns required variables, e.g. runoff . . . 346
mo_mrm_file
Provides file names and units for mRM . . . 348
mo_mrm_global_variables
Global variables for mRM only . . . 353
mo_mrm_init
Wrapper for initializing Routing . . . 362
mo_mrm_mpr
Perform Multiscale Parameter Regionalization on Routing Parameters . . . 369
mo_mrm_net_startup
Startup drainage network for mHM . . . 373
mo_mrm_objective_function_runoff
Objective Functions for Optimization of mHM/mRM against runoff . . . 387

mo_mrm_read_config
Read mRM config . . . 414
mo_mrm_read_data
This module contains all routines to read mRM data from file . . . 417
mo_mrm_restart
Restart routines . . . 422
mo_mrm_river_head . . . 426
mo_mrm_routing
Performs runoff routing for mHM at level L11 . . . 431
mo_mrm_signatures
Module with calculations for several hydrological signatures . . . 439
mo_mrm_write
Write of discharge and restart files . . . 450
mo_mrm_write_fluxes_states
Creates NetCDF output for different fluxes and state variables of mHM . . . 458
mo_ncread . . . 465
mo_ncwrite . . . 482
mo_netcdf
NetCDF Fortran 90 interface wrapper . . . 502
mo_neutrons
Models to predict neutron intensities above soils . . . 562
mo_nml
Deal with namelist files . . . 571
mo_objective_function
Objective Functions for Optimization of mHM . . . 576
mo_optimization_utils . . . 593
mo_orderpack
Sort and ranking routines . . . 593
mo_percentile . . . 610
mo_pet
Module for calculating reference/potential evapotranspiration [mm s-1] . . . 614
mo_read_forcing_nc
Reads forcing input data . . . 622
mo_read_latlon
Reading latitude and longitude coordinates for each basin . . . 629
mo_read_optional_data
Read optional data for mHM calibration . . . 631
mo_read_spatial_data
Reads spatial input data . . . 635
mo_read_timeseries
Routines to read files containing timeseries data . . . 639
mo_restart
Reading and writing states, fluxes and configuration for restart of mHM . . . 641
mo_runoff
Runoff generation for the unsaturated zone, saturated zone (or groundwater zone), and runoff accumulation . . . 645
mo_sce
Shuffled Complex Evolution optimization algorithm . . . 649
mo_set_netcdf_outputs
Defines the structure of the netCDF to write the output in . . . 656
mo_snow_accum_melt
Snow melting and accumulation . . . 657
mo_soil_moisture
Soil moisture of the different layers . . . 659
mo_spatial_agg_disagg_forcing
Spatial aggegation or disaggregation of meteorological input data . . . 664
mo_spatialsimilarity
Routines for bias insensitive comparison of spatial patterns . . . 665

mo_standard_score
Routines for calculating the normalization (anomaly)/standard score/z score and the deseasonalized (standard score on monthly basis) values of a time series . . . 668
mo_startup
Startup procedures for mHM . . . 670
mo_string_utils
String utilities . . . 674
mo_template
Template for future module developments . . . 682
mo_temporal_aggregation
Temporal aggregation for time series (averaging) . . . 685
mo_temporal_disagg_forcing
Temporal disaggregation of daily input values . . . 687
mo_timer
Timing routines . . . 689
mo_utils
General utilities for the CHS library . . . 698
mo_write_ascii
Module to write ascii file output . . . 708
mo_write_fluxes_states
Creates NetCDF output for different fluxes and state variables of mHM . . . 712
mo_xor4096 . . . 722
mo_moment::moment . . . 725
mo_orderpack::mrgref . . . 725
mo_orderpack::mrgrnk . . . 726
mo_errormeasures::mse . . . 726
mo_orderpack::mulcnt . . . 730
mo_percentile::n_element . . . 731
mo_netcdf::ncdataset
Provides basic file modification functionality . . . 732
mo_netcdf::ncdimension
Provides the dimension access functionality . . . 752
mo_netcdf::ncvariable . . . 803
mo_utils::ne . . . 803
mo_orderpack::nearless . . . 803
mo_spatialsimilarity::nndv
Calculates the number of neighboring dominating values, a measure for spatial dissimilarity . . . 804
mo_utils::notequal . . . 806
mo_errormeasures::nse . . . 807
mo_string_utils::num2str
Convert to string . . . 810
mo_string_utils::numarray2str
Convert to string . . . 813
mo_optimization_utils::objective_interface . . . 814
mo_orderpack::omedian . . . 814
mo_write_fluxes_states::outputdataset . . . 815
mo_mrm_write_fluxes_states::outputdataset . . . 819
mo_write_fluxes_states::outputvariable . . . 823
mo_mrm_write_fluxes_states::outputvariable . . . 828
mo_append::paste
Paste (columns) scalars, vectors, and matrixes onto existing array . . . 832
mo_spatialsimilarity::pd
Calculates pattern dissimilarity (PD) measure . . . 835
mo_percentile::percentile . . . 837
mo_common_variables::period . . . 839
mo_percentile::qmedian . . . 841
mo_orderpack::rapknr . . . 842

mo_read_spatial_data::read_spatial_data_ascii
Reads spatial data files of ASCII format . . . 843
mo_corr::realft . . . 845
mo_orderpack::refpar . . . 845
mo_orderpack::refsor . . . 846
mo_orderpack::rinpar . . . 847
mo_errormeasures::rmse . . . 847
mo_orderpack::rnkpar . . . 850
mo_errormeasures::sae . . . 851
mo_julian::setcalendar . . . 854
mo_moment::skewness . . . 855
mo_orderpack::sort
Unconditional ranking . . . 856
mo_orderpack::sort_index . . . 859
mo_spatial_agg_disagg_forcing::spatial_aggregation
Spatial aggregation of meterological variables . . . 861
mo_spatial_agg_disagg_forcing::spatial_disaggregation
Spatial disaggregation of meterological variables . . . 862
mo_utils::special_value
Special IEEE values . . . 863
mo_kind::sprs2_dp
Double Precision Numerical Recipes types for sparse arrays . . . 865
mo_kind::sprs2_sp
Single Precision Numerical Recipes types for sparse arrays . . . 866
mo_errormeasures::sse . . . 867
mo_standard_score::standard_score
Calculates the standard score / normalization (anomaly) / z-score . . . 871
mo_moment::stddev . . . 873
mo_corr::swap . . . 874
mo_utils::swap
Swap to values or two elements in array . . . 874
mo_global_variables::twsstructure . . . 876
mo_orderpack::uniinv . . . 877
mo_orderpack::unipar . . . 877
mo_orderpack::unirnk . . . 878
mo_orderpack::unista . . . 878
mo_restart::unpack_field_and_write
TODO: add description . . . 879
mo_orderpack::valmed . . . 881
mo_orderpack::valnth . . . 882
mo_ncwrite::var2nc
Extended dump_netcdf for multiple variables . . . 883
mo_ncwrite::variable . . . 888
mo_moment::variance . . . 892
mo_errormeasures::wnse . . . 893
mo_xor4096::xor4096 . . . 896
mo_xor4096::xor4096g . . . 897

Chapter 14

File Index

14.1 File List

Here is a list of all files with brief descriptions:

mhm_driver.f90 . . . 899
mo_anneal.f90 . . . 902
mo_append.f90 . . . 902
mo_canopy_interc.f90 . . . 902
mo_common_constants.f90 . . . 902
mo_common_file.f90 . . . 902
mo_common_functions.f90 . . . 903
mo_common_read_config.f90 . . . 903
mo_common_read_data.f90 . . . 903
mo_common_restart.f90 . . . 903
mo_common_variables.f90 . . . 903
mo_constants.f90 . . . 903
mo_corr.f90 . . . 904
mo_dds.f90 . . . 904
mo_errormeasures.f90 . . . 904
mo_file.f90 . . . 904
mo_finish.f90 . . . 905
mo_global_variables.f90 . . . 905
mo_grid.f90 . . . 905
mo_init_states.f90 . . . 905
mo_julian.f90 . . . 905
mo_kind.f90 . . . 905
mo_linfit.f90 . . . 906
mo_mad.f90 . . . 906
mo_mcmc.f90 . . . 906
mo_message.f90 . . . 906
mo_meteo_forcings.f90 . . . 906
mo_mhm.f90 . . . 906
mo_mhm_constants.f90 . . . 907
mo_mhm_eval.f90 . . . 907
mo_mhm_read_config.f90 . . . 907
mo_moment.f90 . . . 907
mo_mrm_constants.f90 . . . 907
mo_mrm_eval.f90 . . . 908
mo_mrm_file.f90 . . . 908
mo_mrm_global_variables.f90 . . . 908
mo_mrm_init.f90 . . . 908
mo_mrm_mpr.f90 . . . 908

mo_mrm_net_startup.f90 . . . 908
mo_mrm_objective_function_runoff.f90 . . . 909
mo_mrm_read_config.f90 . . . 909
mo_mrm_read_data.f90 . . . 909
mo_mrm_restart.f90 . . . 909
mo_mrm_river_head.f90 . . . 910
mo_mrm_routing.f90 . . . 910
mo_mrm_signatures.f90 . . . 910
mo_mrm_write.f90 . . . 910
mo_mrm_write_fluxes_states.f90 . . . 910
mo_ncread.f90 . . . 910
mo_ncwrite.f90 . . . 911
mo_netcdf.f90 . . . 911
mo_neutrons.f90 . . . 911
mo_nml.f90 . . . 911
mo_objective_function.f90 . . . 911
mo_optimization_utils.f90 . . . 912
mo_orderpack.f90 . . . 912
mo_percentile.f90 . . . 912
mo_pet.f90 . . . 913
mo_read_forcing_nc.f90 . . . 913
mo_read_latlon.f90 . . . 913
mo_read_optional_data.f90 . . . 913
mo_read_spatial_data.f90 . . . 913
mo_read_timeseries.f90 . . . 913
mo_restart.f90 . . . 914
mo_runoff.f90 . . . 914
mo_sce.f90 . . . 914
mo_set_netcdf_outputs.f90 . . . 916
mo_snow_accum_melt.f90 . . . 916
mo_soil_moisture.f90 . . . 916
mo_spatial_agg_disagg_forcing.f90 . . . 916
mo_spatialsimilarity.f90 . . . 917
mo_standard_score.f90 . . . 917
mo_startup.f90 . . . 917
mo_string_utils.f90 . . . 917
mo_template.f90 . . . 917
mo_temporal_aggregation.f90 . . . 918
mo_temporal_disagg_forcing.f90 . . . 918
mo_timer.f90 . . . 918
mo_utils.f90 . . . 918
mo_write_ascii.f90 . . . 919
mo_write_fluxes_states.f90 . . . 919
mo_xor4096.f90 . . . 919
mrm_driver.f90 . . . 919

Chapter 15

Data Type Documentation

15.1 mo_moment::absdev Interface Reference

Public Member Functions

- real(sp) function absdev_sp (dat, mask)
- real(dp) function absdev_dp (dat, mask)

15.1.1 Detailed Description

Definition at line 110 of file mo_moment.f90.

15.1.2 Member Function/Subroutine Documentation

15.1.2.1 real(dp) function mo_moment::absdev::absdev_dp (real(dp), dimension(:), intent(in) *dat,* logical, dimension(:), intent(in), optional *mask*)

Definition at line 849 of file mo_moment.f90.

Referenced by mo_moment::absdev_dp().

Here is the caller graph for this function:

15.1.2.2 real(sp) function mo_moment::absdev::absdev_sp (real(sp), dimension(:), intent(in) *dat,* logical, dimension(:), intent(in), optional *mask*)

Definition at line 880 of file mo_moment.f90.

Referenced by mo_moment::absdev_sp().

Here is the caller graph for this function:

The documentation for this interface was generated from the following file:

- mo_moment.f90

15.2 mo_anneal::anneal Interface Reference

anneal

Public Member Functions

- real(dp) function, dimension(size(para, 1)) anneal_dp (eval, cost, para, prange, prange_func, temp, Dt, nl↩TERmax, Len, nST, eps, acc, seeds, printflag, maskpara, weight, changeParaMode, reflectionFlag, pertub↩FlexFlag, maxit, undef_funcval, tmp_file, funcbest, history)

15.2.1 Detailed Description

anneal

Optimizes a user provided cost function using the Simulated Annealing strategy.

Parameters

in	*INTERFACE :: cost_dp*	interface calculating the cost function at a given point
in	*REAL(DP), DIMENSION(:) :: para*	initial parameter set
in	*REAL(DP), DIMENSIO↩N(size(para),2), optional :: prange*	lower and upper bound per parameter
in	*INTERFACE, optional :: prange_func*	interface calculating the feasible range for a parameter at a certain point, if ranges are variable

in	*REAL(DP), optional :: temp*	initial temperature DEFAULT: Get_Temperature
in	*REAL(DP), optional :: DT*	geometrical decreement of temperature 0.7<DT<0.999 DEFAULT: 0.9_dp
in	*INTEGER(I4), optional :: nITERmax*	maximal number of iterations will be increased by 10% if stopping criteria of acceptance ratio or epsilon decreement of cost function is not fullfilled DEFAULT: 1000_i4
in	*INTEGER(I4), optional :: LEN*	Length of Markov Chain DEFAULT: MAX(250_i4, size(para,1))
in	*INTEGER(I4), optional :: nST*	Number of consecutive LEN steps DEFAULT: 5_i4
in	*REAL(DP), optional :: eps*	Stopping criteria of epsilon decreement of cost function DEFAULT: 0.0001_dp
in	*REAL(DP), optional :: acc*	Stopping criteria for Acceptance Ratio acc <= 0.1_dp DEFAULT: 0.1_dp
in	*INTEGER(I4/I8), DIMENSION(3), optional :: seeds*	Seeds of random numbers used for random parameter set generation DEFAULT: dependent on current time
in	*LOGICAL, optional :: printflag*	If .true. detailed command line output is written DEFAULT: .false.
in	*LOGICAL, DIMENSIO↩N(size(para)), optional :: maskpara*	maskpara(i) = .true. –> parameter is optimized maskpara(i) = .false. –> parameter is discarded from optimiztaion DEFAULT: .true.
in	*REAL(DP), DIMENSIO↩N(size(para)), optional :: weight*	vector of weights per parameter: gives the frequency of parameter to be chosen for optimization (will be scaled to a CDF internally) eg. [1,2,1] –> parameter 2 is chosen twice as often as parameter 1 and 2 DEFAULT: weight = 1.0_dp
in	*INTEGER(I4), optional :: changePara↩Mode*	which and how many param.are changed in one step 1 = one parameter 2 = all parameter 3 = neighborhood parameter DEFAULT: 1_i4
in	*LOGICAL, optional :: reflectionFlag*	if new parameter values are Gaussian distributed and reflected (.true.) or uniform in range (.false.) DEFAULT: .false.
in	*LOGICAL, optional :: pertubFlexFlag*	if pertubation of Gaussian distributed parameter values is constant at 0.↩2 (.false.) or depends on dR (.true.) DEFAULT: .true.
in	*LOGICAL, optional :: maxit*	maximizing (.true.) or minimizing (.false.) a function DEFAULT: .false. (minimization)
in	*REAL(DP), optional :: undef_funcval*	objective function value defining invalid model output, e.g. -999.0_dp

in	*character(len=∗) , optional :: tmp_file*	file with temporal output
out	*REAL(DP), optional :: funcbest*	minimized value of cost function
out	*"REAL(DP),DI↩ MENSION(,↩ :),ALLOCATA↩ BLE,optional*	:: history" returns a vector of achieved objective

Returns

real(dp) :: parabest(size(para)) — Parameter set minimizing the cost function.

Note

Either fixed parameter range (prange) OR flexible parameter range (function interface prange_func) has to be given in calling sequence.
Only double precision version available.
If single precision is needed not only DP has to be replaced by SP but also I8 of save_state (random number variables) has to be replaced by I4.
ParaChangeMode $>$ 1 is not applied in GetTemperature. For Temperature estimation always only one single parameter is changed (ParaChangeMode=1) which should give theoretically always the best estimate.
Cost and prange_func are user defined functions. See interface definition.

Definition at line 225 of file mo_anneal.f90.

15.2.2 Member Function/Subroutine Documentation

15.2.2.1 real(dp) function, dimension(size(para, 1)) mo_anneal::anneal::anneal_dp (procedure(eval_interface), intent(in), pointer *eval,* procedure(objective_interface), intent(in), pointer *cost,* real(dp), dimension(:), intent(in) *para,* real(dp), dimension(size(para, 1), 2), intent(in), optional *prange,* optional *prange_func,* real(dp), intent(in), optional *temp,* real(dp), intent(in), optional *Dt,* integer(i4), intent(in), optional *nITERmax,* integer(i4), intent(in), optional *Len,* integer(i4), intent(in), optional *nST,* real(dp), intent(in), optional *eps,* real(dp), intent(in), optional *acc,* integer(i8), dimension(3), intent(in), optional *seeds,* logical, intent(in), optional *printflag,* logical, dimension(size(para, 1)), intent(in), optional *maskpara,* real(dp), dimension(size(para, 1)), intent(in), optional *weight,* integer(i4), intent(in), optional *changeParaMode,* logical, intent(in), optional *reflectionFlag,* logical, intent(in), optional *pertubFlexFlag,* logical, intent(in), optional *maxit,* real(dp), intent(in), optional *undef_funcval,* character(len = ∗), intent(in), optional *tmp_file,* real(dp), intent(out), optional *funcbest,* real(dp), dimension(:, :), intent(out), optional, allocatable *history*)

Definition at line 357 of file mo_anneal.f90.

The documentation for this interface was generated from the following file:

- mo_anneal.f90

15.3 mo_append::append Interface Reference

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

Public Member Functions

- subroutine append_i4_v_s (vec1, sca2)
- subroutine append_i4_v_v (vec1, vec2)

- subroutine append_i4_m_m (mat1, mat2, fill_value)
- subroutine append_i8_v_s (vec1, sca2)
- subroutine append_i8_v_v (vec1, vec2)
- subroutine append_i8_m_m (mat1, mat2, fill_value)
- subroutine append_i8_3d (mat1, mat2, fill_value)
- subroutine append_sp_v_s (vec1, sca2)
- subroutine append_sp_v_v (vec1, vec2)
- subroutine append_sp_m_m (mat1, mat2, fill_value)
- subroutine append_sp_3d (mat1, mat2, fill_value)
- subroutine append_dp_v_s (vec1, sca2)
- subroutine append_dp_v_v (vec1, vec2)
- subroutine append_dp_m_m (mat1, mat2, fill_value)
- subroutine append_dp_3d (mat1, mat2, fill_value)
- subroutine append_char_v_s (vec1, sca2)
- subroutine append_char_v_v (vec1, vec2)
- subroutine append_char_m_m (mat1, mat2, fill_value)
- subroutine append_char_3d (mat1, mat2, fill_value)
- subroutine append_lgt_v_s (vec1, sca2)
- subroutine append_lgt_v_v (vec1, vec2)
- subroutine append_lgt_m_m (mat1, mat2, fill_value)
- subroutine append_lgt_3d (mat1, mat2, fill_value)

15.3.1 Detailed Description

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

Appends one input to the rows of another, i.e. append on the first dimension.
The input might be a scalar, a vector or a matrix.
Possibilities are:
(1) append scalar to vector
(2) append vector to vector
(3) append matrix to matrix

Parameters

in	*input2*	values to append. Can be INTEGER(I4/I8), REAL(SP/DP), or CHARACTE↩R(len=∗) and also scalar, DIMENSION(:), or DIMENSION(:,:) Also includes 3d version, where the append is always done along the first dimension. If not scalar then the columns have to agree with input1.
in,out	*allocatable :: input1*	array to be appended to. Can be INTEGER(I4/I8), REAL(SP/DP), or CHAR↩ACTER(len=∗). Must be DIMENSION(:) or DIMENSION(:,:), and allocatable. If input2 is not scalar then it must be size(input1,2) = size(input2,2).

Author

Juliane Mai

Date

Aug 2012

Definition at line 116 of file mo_append.f90.

15.3.2 Member Function/Subroutine Documentation

15.3.2.1 subroutine mo_append::append::append_char_3d (character(len = ∗), dimension(:, :, :), intent(inout), allocatable *mat1,* character(len = ∗), dimension(:, :, :), intent(in) *mat2,* character(len = ∗), intent(in), optional *fill_value*)

Definition at line 1022 of file mo_append.f90.

15.3.2.2 subroutine mo_append::append::append_char_m_m (character(len = ∗), dimension(:, :), intent(inout), allocatable *mat1,* character(len = ∗), dimension(:, :), intent(in) *mat2,* character(len = ∗), intent(in), optional *fill_value*)

Definition at line 963 of file mo_append.f90.

15.3.2.3 subroutine mo_append::append::append_char_v_s (character(len = ∗), dimension(:), intent(inout), allocatable *vec1,* character(len = ∗), intent(in) *sca2*)

Definition at line 901 of file mo_append.f90.

15.3.2.4 subroutine mo_append::append::append_char_v_v (character(len = ∗), dimension(:), intent(inout), allocatable *vec1,* character(len = ∗), dimension(:), intent(in) *vec2*)

Definition at line 933 of file mo_append.f90.

15.3.2.5 subroutine mo_append::append::append_dp_3d (real(dp), dimension(:, :, :), intent(inout), allocatable *mat1,* real(dp), dimension(:, :, :), intent(in) *mat2,* real(dp), intent(in), optional *fill_value*)

Definition at line 852 of file mo_append.f90.

15.3.2.6 subroutine mo_append::append::append_dp_m_m (real(dp), dimension(:, :), intent(inout), allocatable *mat1,* real(dp), dimension(:, :), intent(in) *mat2,* real(dp), intent(in), optional *fill_value*)

Definition at line 793 of file mo_append.f90.

15.3.2.7 subroutine mo_append::append::append_dp_v_s (real(dp), dimension(:), intent(inout), allocatable *vec1,* real(dp), intent(in) *sca2*)

Definition at line 729 of file mo_append.f90.

15.3.2.8 subroutine mo_append::append::append_dp_v_v (real(dp), dimension(:), intent(inout), allocatable *vec1,* real(dp), dimension(:), intent(in) *vec2*)

Definition at line 761 of file mo_append.f90.

15.3.2.9 subroutine mo_append::append::append_i4_m_m (integer(i4), dimension(:, :), intent(inout), allocatable *mat1,* integer(i4), dimension(:, :), intent(in) *mat2,* integer(i4), intent(in), optional *fill_value*)

Definition at line 277 of file mo_append.f90.

15.3.2.10 subroutine mo_append::append::append_i4_v_s (integer(i4), dimension(:), intent(inout), allocatable *vec1,* integer(i4), intent(in) *sca2*)

Definition at line 213 of file mo_append.f90.

15.3.2.11 subroutine mo_append::append::append_i4_v_v (integer(i4), dimension(:), intent(inout), allocatable *vec1,* integer(i4), dimension(:), intent(in) *vec2*)

Definition at line 245 of file mo_append.f90.

15.3.2.12 subroutine mo_append::append::append_i8_3d (integer(i8), dimension(:, :, :), intent(inout), allocatable *mat1,* integer(i8), dimension(:, :, :), intent(in) *mat2,* integer(i8), intent(in), optional *fill_value*)

Definition at line 508 of file mo_append.f90.

15.3.2.13 subroutine mo_append::append::append_i8_m_m (integer(i8), dimension(:, :), intent(inout), allocatable *mat1,* integer(i8), dimension(:, :), intent(in) *mat2,* integer(i8), intent(in), optional *fill_value*)

Definition at line 449 of file mo_append.f90.

15.3.2.14 subroutine mo_append::append::append_i8_v_s (integer(i8), dimension(:), intent(inout), allocatable *vec1,* integer(i8), intent(in) *sca2*)

Definition at line 385 of file mo_append.f90.

15.3.2.15 subroutine mo_append::append::append_i8_v_v (integer(i8), dimension(:), intent(inout), allocatable *vec1,* integer(i8), dimension(:), intent(in) *vec2*)

Definition at line 417 of file mo_append.f90.

15.3.2.16 subroutine mo_append::append::append_lgt_3d (logical, dimension(:, :, :), intent(inout), allocatable *mat1,* logical, dimension(:, :, :), intent(in) *mat2,* logical, intent(in), optional *fill_value*)

Definition at line 1194 of file mo_append.f90.

15.3.2.17 subroutine mo_append::append::append_lgt_m_m (logical, dimension(:, :), intent(inout), allocatable *mat1,* logical, dimension(:, :), intent(in) *mat2,* logical, intent(in), optional *fill_value*)

Definition at line 1135 of file mo_append.f90.

15.3.2.18 subroutine mo_append::append::append_lgt_v_s (logical, dimension(:), intent(inout), allocatable *vec1,* logical, intent(in) *sca2*)

Definition at line 1071 of file mo_append.f90.

15.3.2.19 subroutine mo_append::append::append_lgt_v_v (logical, dimension(:), intent(inout), allocatable *vec1,* logical, dimension(:), intent(in) *vec2*)

Definition at line 1103 of file mo_append.f90.

15.3.2.20 subroutine mo_append::append::append_sp_3d (real(sp), dimension(:, :, :), intent(inout), allocatable *mat1,* real(sp), dimension(:, :, :), intent(in) *mat2,* real(sp), intent(in), optional *fill_value*)

Definition at line 680 of file mo_append.f90.

15.3.2.21 subroutine mo_append::append::append_sp_m_m (real(sp), dimension(:, :), intent(inout), allocatable *mat1,* real(sp), dimension(:, :), intent(in) *mat2,* real(sp), intent(in), optional *fill_value*)

Definition at line 621 of file mo_append.f90.

15.3.2.22 subroutine mo_append::append::append_sp_v_s (real(sp), dimension(:), intent(inout), allocatable *vec1,* real(sp), intent(in) *sca2*)

Definition at line 557 of file mo_append.f90.

15.3.2.23 subroutine mo_append::append::append_sp_v_v (real(sp), dimension(:), intent(inout), allocatable *vec1,* real(sp), dimension(:), intent(in) *vec2*)

Definition at line 589 of file mo_append.f90.

The documentation for this interface was generated from the following file:

- mo_append.f90

15.4 mo_corr::arth Interface Reference

Private Member Functions

- real(sp) function, dimension(n) arth_sp (first, increment, n)
- real(dp) function, dimension(n) arth_dp (first, increment, n)
- integer(i4) function, dimension(n) arth_i4 (first, increment, n)

15.4.1 Detailed Description

Definition at line 377 of file mo_corr.f90.

15.4.2 Member Function/Subroutine Documentation

15.4.2.1 real(dp) function, dimension(n) mo_corr::arth::arth_dp (real(dp), intent(in) *first,* real(dp), intent(in) *increment,* integer(i4), intent(in) *n*) [private]

Definition at line 453 of file mo_corr.f90.

Referenced by mo_corr::arth_dp().

Here is the caller graph for this function:

15.4.2.2 integer(i4) function, dimension(n) mo_corr::arth::arth_i4 (integer(i4), intent(in) *first,* integer(i4), intent(in) *increment,* integer(i4), intent(in) *n*) [private]

Definition at line 487 of file mo_corr.f90.

Referenced by mo_corr::arth_i4().

Here is the caller graph for this function:

15.4.2.3 real(sp) function, dimension(n) mo_corr::arth::arth_sp (real(sp), intent(in) *first*, real(sp), intent(in) *increment*, integer(i4), intent(in) *n*) [private]

Definition at line 419 of file mo_corr.f90.

Referenced by mo_corr::arth_sp().

Here is the caller graph for this function:

The documentation for this interface was generated from the following file:

- mo_corr.f90

15.5 mo_ncwrite::attribute Type Reference

Collaboration diagram for mo_ncwrite::attribute:

Public Attributes

- character(len=maxlen) name
- integer(i4) xtype
- integer(i4) nvalues
- character(len=maxlen) values

15.5.1 Detailed Description

Definition at line 65 of file mo_ncwrite.f90.

15.5.2 Member Data Documentation

15.5.2.1 character (len = maxlen) mo_ncwrite::attribute::name

Definition at line 66 of file mo_ncwrite.f90.

15.5.2.2 integer(i4) mo_ncwrite::attribute::nvalues

Definition at line 68 of file mo_ncwrite.f90.

15.5.2.3 character (len = maxlen) mo_ncwrite::attribute::values

Definition at line 69 of file mo_ncwrite.f90.

15.5.2.4 integer(i4) mo_ncwrite::attribute::xtype

Definition at line 67 of file mo_ncwrite.f90.

The documentation for this type was generated from the following file:

- mo_ncwrite.f90

15.6 mo_corr::autocoeffk Interface Reference

Public Member Functions

- real(sp) function autocoeffk_sp (x, k, mask)
- real(dp) function autocoeffk_dp (x, k, mask)
- real(dp) function, dimension(size(k)) autocoeffk_1d_dp (x, k, mask)
- real(sp) function, dimension(size(k)) autocoeffk_1d_sp (x, k, mask)

15.6.1 Detailed Description

Definition at line 127 of file mo_corr.f90.

15.6.2 Member Function/Subroutine Documentation

15.6.2.1 real(dp) function, dimension(size(k)) mo_corr::autocoeffk::autocoeffk_1d_dp (real(dp), dimension(:), intent(in) *x*, integer(i4), dimension(:), intent(in) *k*, logical, dimension(:), intent(in), optional *mask*)

Definition at line 557 of file mo_corr.f90.

15.6.2.2 real(sp) function, dimension(size(k)) mo_corr::autocoeffk::autocoeffk_1d_sp (real(sp), dimension(:), intent(in) *x*, integer(i4), dimension(:), intent(in) *k*, logical, dimension(:), intent(in), optional *mask*)

Definition at line 581 of file mo_corr.f90.

15.6.2.3 real(dp) function mo_corr::autocoeffk::autocoeffk_dp (real(dp), dimension(:), intent(in) *x*, integer(i4), intent(in) *k*, logical, dimension(:), intent(in), optional *mask*)

Definition at line 520 of file mo_corr.f90.

Referenced by mo_corr::autocoeffk_dp().

Here is the caller graph for this function:

15.6.2.4 real(sp) function mo_corr::autocoeffk::autocoeffk_sp (real(sp), dimension(:), intent(in) *x*, integer(i4), intent(in) *k*, logical, dimension(:), intent(in), optional *mask*)

Definition at line 539 of file mo_corr.f90.

Referenced by mo_corr::autocoeffk_sp().

Here is the caller graph for this function:

The documentation for this interface was generated from the following file:

- mo_corr.f90

15.7 mo_corr::autocorr Interface Reference

Public Member Functions

- real(sp) function autocorr_sp (x, k, mask)

- real(dp) function autocorr_dp (x, k, mask)
- real(sp) function, dimension(size(k)) autocorr_1d_sp (x, k, mask)
- real(dp) function, dimension(size(k)) autocorr_1d_dp (x, k, mask)

15.7.1 Detailed Description

Definition at line 184 of file mo_corr.f90.

15.7.2 Member Function/Subroutine Documentation

15.7.2.1 real(dp) function, dimension(size(k)) mo_corr::autocorr::autocorr_1d_dp (real(dp), dimension(:), intent(in) *x*, integer(i4), dimension(:), intent(in) *k*, logical, dimension(:), intent(in), optional *mask*)

Definition at line 643 of file mo_corr.f90.

15.7.2.2 real(sp) function, dimension(size(k)) mo_corr::autocorr::autocorr_1d_sp (real(sp), dimension(:), intent(in) *x*, integer(i4), dimension(:), intent(in) *k*, logical, dimension(:), intent(in), optional *mask*)

Definition at line 669 of file mo_corr.f90.

15.7.2.3 real(dp) function mo_corr::autocorr::autocorr_dp (real(dp), dimension(:), intent(in) *x*, integer(i4), intent(in) *k*, logical, dimension(:), intent(in), optional *mask*)

Definition at line 606 of file mo_corr.f90.

Referenced by mo_corr::autocorr_dp().

Here is the caller graph for this function:

15.7.2.4 real(sp) function mo_corr::autocorr::autocorr_sp (real(sp), dimension(:), intent(in) *x*, integer(i4), intent(in) *k*, logical, dimension(:), intent(in), optional *mask*)

Definition at line 625 of file mo_corr.f90.

Referenced by mo_corr::autocorr_sp().

Here is the caller graph for this function:

The documentation for this interface was generated from the following file:

- mo_corr.f90

15.8 mo_moment::average Interface Reference

Public Member Functions

- real(sp) function average_sp (dat, mask)
- real(dp) function average_dp (dat, mask)

15.8.1 Detailed Description

Definition at line 163 of file mo_moment.f90.

15.8.2 Member Function/Subroutine Documentation

15.8.2.1 real(dp) function mo_moment::average::average_dp (real(dp), dimension(:), intent(in) *dat*, logical, dimension(:), intent(in), optional *mask*)

Definition at line 912 of file mo_moment.f90.

Referenced by mo_moment::average_dp().

Here is the caller graph for this function:

15.8.2.2 real(sp) function mo_moment::average::average_sp (real(sp), dimension(:), intent(in) *dat,* logical, dimension(:), intent(in), optional *mask*)

Definition at line 938 of file mo_moment.f90.

Referenced by mo_moment::average_sp().

Here is the caller graph for this function:

The documentation for this interface was generated from the following file:

- mo_moment.f90

15.9 mo_mrm_global_variables::basininfo_mrm Type Reference

Collaboration diagram for mo_mrm_global_variables::basininfo_mrm:

Public Attributes

- integer(i4) ngauges
- integer(i4), dimension(:), allocatable gaugeidlist
- integer(i4), dimension(:), allocatable gaugeindexlist
- integer(i4), dimension(:), allocatable gaugenodelist
- integer(i4) ninflowgauges
- integer(i4), dimension(:), allocatable inflowgaugeidlist
- integer(i4), dimension(:), allocatable inflowgaugeindexlist
- integer(i4), dimension(:), allocatable inflowgaugenodelist
- logical, dimension(:), allocatable inflowgaugeheadwater
- integer(i4) l0_noutlet
- integer(i4), dimension(:), allocatable l0_rowoutlet
- integer(i4), dimension(:), allocatable l0_coloutlet

15.9.1 Detailed Description

Definition at line 89 of file mo_mrm_global_variables.f90.

15.9.2 Member Data Documentation

15.9.2.1 integer(i4), dimension(:), allocatable mo_mrm_global_variables::basininfo_mrm::gaugeidlist

Definition at line 93 of file mo_mrm_global_variables.f90.

15.9.2.2 integer(i4), dimension(:), allocatable mo_mrm_global_variables::basininfo_mrm::gaugeindexlist

Definition at line 94 of file mo_mrm_global_variables.f90.

15.9.2.3 integer(i4), dimension(:), allocatable mo_mrm_global_variables::basininfo_mrm::gaugenodelist

Definition at line 95 of file mo_mrm_global_variables.f90.

15.9.2.4 logical, dimension(:), allocatable mo_mrm_global_variables::basininfo_mrm::inflowgaugeheadwater

Definition at line 102 of file mo_mrm_global_variables.f90.

15.9.2.5 integer(i4), dimension(:), allocatable mo_mrm_global_variables::basininfo_mrm::inflowgaugeidlist

Definition at line 99 of file mo_mrm_global_variables.f90.

15.9.2.6 integer(i4), dimension(:), allocatable mo_mrm_global_variables::basininfo_mrm::inflowgaugeindexlist

Definition at line 100 of file mo_mrm_global_variables.f90.

15.9.2.7 integer(i4), dimension(:), allocatable mo_mrm_global_variables::basininfo_mrm::inflowgaugenodelist

Definition at line 101 of file mo_mrm_global_variables.f90.

15.9.2.8 integer(i4), dimension(:), allocatable mo_mrm_global_variables::basininfo_mrm::l0_coloutlet

Definition at line 108 of file mo_mrm_global_variables.f90.

15.9.2.9 integer(i4) mo_mrm_global_variables::basininfo_mrm::l0_noutlet

Definition at line 106 of file mo_mrm_global_variables.f90.

15.9.2.10 integer(i4), dimension(:), allocatable mo_mrm_global_variables::basininfo_mrm::l0_rowoutlet

Definition at line 107 of file mo_mrm_global_variables.f90.

15.9.2.11 integer(i4) mo_mrm_global_variables::basininfo_mrm::ngauges

Definition at line 92 of file mo_mrm_global_variables.f90.

15.9.2.12 integer(i4) mo_mrm_global_variables::basininfo_mrm::ninflowgauges

Definition at line 98 of file mo_mrm_global_variables.f90.

The documentation for this type was generated from the following file:

- mo_mrm_global_variables.f90

15.10 mo_errormeasures::bias Interface Reference

Public Member Functions

- real(sp) function bias_sp_1d (x, y, mask)
- real(dp) function bias_dp_1d (x, y, mask)
- real(sp) function bias_sp_2d (x, y, mask)
- real(dp) function bias_dp_2d (x, y, mask)
- real(sp) function bias_sp_3d (x, y, mask)
- real(dp) function bias_dp_3d (x, y, mask)

15.10.1 Detailed Description

Definition at line 104 of file mo_errormeasures.f90.

15.10.2 Member Function/Subroutine Documentation

15.10.2.1 real(dp) function mo_errormeasures::bias::bias_dp_1d (real(dp), dimension(:), intent(in) *x*, real(dp), dimension(:), intent(in) *y*, logical, dimension(:), intent(in), optional *mask*)

Definition at line 819 of file mo_errormeasures.f90.

Referenced by mo_errormeasures::bias_dp_1d().

Here is the caller graph for this function:

15.10.2.2 real(dp) function mo_errormeasures::bias::bias_dp_2d (real(dp), dimension(:, :), intent(in) *x*, real(dp), dimension(:, :), intent(in) *y*, logical, dimension(:, :), intent(in), optional *mask*)

Definition at line 896 of file mo_errormeasures.f90.

Referenced by mo_errormeasures::bias_dp_2d().

Here is the caller graph for this function:

15.10.2.3 real(dp) function mo_errormeasures::bias::bias_dp_3d (real(dp), dimension(:, :, :), intent(in) *x*, real(dp), dimension(:, :, :), intent(in) *y*, logical, dimension(:, :, :), intent(in), optional *mask*)

Definition at line 978 of file mo_errormeasures.f90.

Referenced by mo_errormeasures::bias_dp_3d().

Here is the caller graph for this function:

15.10.2.4 real(sp) function mo_errormeasures::bias::bias_sp_1d (real(sp), dimension(:), intent(in) *x*, real(sp), dimension(:), intent(in) *y*, logical, dimension(:), intent(in), optional *mask*)

Definition at line 782 of file mo_errormeasures.f90.

Referenced by mo_errormeasures::bias_sp_1d().

Here is the caller graph for this function:

15.10.2.5 real(sp) function mo_errormeasures::bias::bias_sp_2d (real(sp), dimension(:, :), intent(in) *x*, real(sp), dimension(:, :), intent(in) *y*, logical, dimension(:, :), intent(in), optional *mask*)

Definition at line 855 of file mo_errormeasures.f90.

Referenced by mo_errormeasures::bias_sp_2d().

Here is the caller graph for this function:

15.10.2.6 real(sp) function mo_errormeasures::bias::bias_sp_3d (real(sp), dimension(:, :, :), intent(in) *x*, real(sp), dimension(:, :, :), intent(in) *y*, logical, dimension(:, :, :), intent(in), optional *mask*)

Definition at line 936 of file mo_errormeasures.f90.

Referenced by mo_errormeasures::bias_sp_3d().

Here is the caller graph for this function:

The documentation for this interface was generated from the following file:

- mo_errormeasures.f90

15.11 mo_moment::central_moment Interface Reference

Public Member Functions

- real(sp) function central_moment_sp (x, r, mask)
- real(dp) function central_moment_dp (x, r, mask)

15.11.1 Detailed Description

Definition at line 217 of file mo_moment.f90.

15.11.2 Member Function/Subroutine Documentation

15.11.2.1 real(dp) function mo_moment::central_moment::central_moment_dp (real(dp), dimension(:), intent(in) *x*, integer(i4), intent(in) *r*, logical, dimension(:), intent(in), optional *mask*)

Definition at line 965 of file mo_moment.f90.

Referenced by mo_moment::central_moment_dp().

Here is the caller graph for this function:

15.11.2.2 real(sp) function mo_moment::central_moment::central_moment_sp (real(sp), dimension(:), intent(in) *x*, integer(i4), intent(in) *r*, logical, dimension(:), intent(in), optional *mask*)

Definition at line 1004 of file mo_moment.f90.

Referenced by mo_moment::central_moment_sp().

Here is the caller graph for this function:

The documentation for this interface was generated from the following file:

- mo_moment.f90

15.12 mo_moment::central_moment_var Interface Reference

Public Member Functions

- real(sp) function central_moment_var_sp (x, r, mask)
- real(dp) function central_moment_var_dp (x, r, mask)

15.12.1 Detailed Description

Definition at line 271 of file mo_moment.f90.

15.12.2 Member Function/Subroutine Documentation

15.12.2.1 real(dp) function mo_moment::central_moment_var::central_moment_var_dp (real(dp), dimension(:), intent(in) *x*, integer(i4), intent(in) *r*, logical, dimension(:), intent(in), optional *mask*)

Definition at line 1044 of file mo_moment.f90.

Referenced by mo_moment::central_moment_var_dp().

Here is the caller graph for this function:

15.12.2.2 real(sp) function mo_moment::central_moment_var::central_moment_var_sp (real(sp), dimension(:), intent(in) *x*, integer(i4), intent(in) *r*, logical, dimension(:), intent(in), optional *mask*)

Definition at line 1082 of file mo_moment.f90.

Referenced by mo_moment::central_moment_var_sp().

Here is the caller graph for this function:

The documentation for this interface was generated from the following file:

- mo_moment.f90

15.13 mo_standard_score::classified_standard_score Interface Reference

Calculates the classified standard score (e.g. classes are months).

Public Member Functions

- real(sp) function, dimension(size(data, dim=1)) classified_standard_score_sp (data, classes, mask)
- real(dp) function, dimension(size(data, dim=1)) classified_standard_score_dp (data, classes, mask)

15.13.1 Detailed Description

Calculates the classified standard score (e.g. classes are months).

In statistics, the standard score is the (signed) number of standard deviations an observation or datum is above the mean. Thus, a positive standard score indicates a datum above the mean, while a negative standard score indicates a datum below the mean. It is a dimensionless quantity obtained by subtracting the population mean from an individual raw score and then dividing the difference by the population standard deviation. This conversion process is called standardizing or normalizing (however, "normalizing" can refer to many types of ratios).
Standard scores are also called z-values, z-scores, normal scores, and standardized variables; the use of "Z" is because the normal distribution is also known as the "Z distribution". They are most frequently used to compare a sample to a standard normal deviate, though they can be defined without assumptions of normality (Wikipedia, May 2015).
In this particular case the standard score is calculated for means and standard deviations derived from classes of the time series. Such classes could be for example months. Thus, the output would be a deseasonalized time series.

$$classified_standard_score = \frac{x_i - \mu_{c_{x_i}}}{\sigma_{c_{x_i}}}$$

where x_i is an element of class c_{x_i}. x is a population, $\mu_{c_{x_i}}$ is the mean of all members of a class c_{x_i} and $\sigma_{c_{x_i}}$ its standard deviation.

If an optinal mask is given, the calculations are over those locations that correspond to true values in the mask. data can be single or double precision. The result will have the same numerical precision.

Parameters

in	*integer, dimension(:) :: classes*	classes to categorize data (e.g. months)
in	*real(sp/dp), dimension(:) :: data*	data to calculate the standard score for
in	*logical, dimension(:), optional :: mask*	indication which cells to use for calculation If present, only those locations in mask having true values in mask are evaluated.

Returns

real(sp/dp) :: classified_standard_score — classified standard score (e.g. deseasonalized time series)

Author

Matthias Zink

Date

May 2015

Definition at line 191 of file mo_standard_score.f90.

15.13.2 Member Function/Subroutine Documentation

15.13.2.1 real(dp) function, dimension(size(data, dim = 1)) mo_standard_score::classified_standard_score::classified_standard_score_dp (real(dp), dimension(:), intent(in) *data*, integer, dimension(:), intent(in) *classes*, logical, dimension(:), intent(in), optional *mask*)

Definition at line 321 of file mo_standard_score.f90.

Referenced by mo_standard_score::classified_standard_score_dp().

Here is the caller graph for this function:

15.13.2.2 real(sp) function, dimension(size(data, dim = 1)) mo_standard_score::classified_standard_score::classified_standard_score_sp (real(sp), dimension(:), intent(in) *data,* integer, dimension(:), intent(in) *classes,* logical, dimension(:), intent(in), optional *mask*)

Definition at line 263 of file mo_standard_score.f90.

Referenced by mo_standard_score::classified_standard_score_sp().

Here is the caller graph for this function:

The documentation for this interface was generated from the following file:

- mo_standard_score.f90

15.14 mo_corr::corr Interface Reference

Public Member Functions

- real(sp) function, dimension(size(data1)) corr_sp (data1, data2, nadjust, nhigh, nwin)
- real(dp) function, dimension(size(data1)) corr_dp (data1, data2, nadjust, nhigh, nwin)

15.14.1 Detailed Description

Definition at line 253 of file mo_corr.f90.

15.14.2 Member Function/Subroutine Documentation

15.14.2.1 real(dp) function, dimension(size(data1)) mo_corr::corr::corr_dp (real(dp), dimension(:), intent(in) *data1,* real(dp), dimension(:), intent(in) *data2,* integer(i4), intent(out), optional *nadjust,* integer(i4), intent(in), optional *nhigh,* integer(i4), intent(in), optional *nwin*)

Definition at line 697 of file mo_corr.f90.

Referenced by mo_corr::corr_dp().

Here is the caller graph for this function:

15.14.2.2 real(sp) function, dimension(size(data1)) mo_corr::corr::corr_sp (real(sp), dimension(:), intent(in) *data1,* real(sp), dimension(:), intent(in) *data2,* integer(i4), intent(out), optional *nadjust,* integer(i4), intent(in), optional *nhigh,* integer(i4), intent(in), optional *nwin*)

Definition at line 806 of file mo_corr.f90.

Referenced by mo_corr::corr_sp().

Here is the caller graph for this function:

The documentation for this interface was generated from the following file:

- mo_corr.f90

15.15 mo_moment::correlation Interface Reference

Public Member Functions

- real(sp) function correlation_sp (x, y, mask)
- real(dp) function correlation_dp (x, y, mask)

15.15.1 Detailed Description

Definition at line 327 of file mo_moment.f90.

15.15.2 Member Function/Subroutine Documentation

15.15.2.1 real(dp) function mo_moment::correlation::correlation_dp (real(dp), dimension(:), intent(in) *x,* real(dp), dimension(:), intent(in) *y,* logical, dimension(:), intent(in), optional *mask*)

Definition at line 1121 of file mo_moment.f90.

Referenced by mo_moment::correlation_dp().

Here is the caller graph for this function:

15.15.2.2 real(sp) function mo_moment::correlation::correlation_sp (real(sp), dimension(:), intent(in) *x*, real(sp), dimension(:), intent(in) *y*, logical, dimension(:), intent(in), optional *mask*)

Definition at line 1156 of file mo_moment.f90.

Referenced by mo_moment::correlation_sp().

Here is the caller graph for this function:

The documentation for this interface was generated from the following file:

- mo_moment.f90

15.16 mo_moment::covariance Interface Reference

Public Member Functions

- real(sp) function covariance_sp (x, y, mask)
- real(dp) function covariance_dp (x, y, mask)

15.16.1 Detailed Description

Definition at line 383 of file mo_moment.f90.

15.16.2 Member Function/Subroutine Documentation

15.16.2.1 real(dp) function mo_moment::covariance::covariance_dp (real(dp), dimension(:), intent(in) *x*, real(dp), dimension(:), intent(in) *y*, logical, dimension(:), intent(in), optional *mask*)

Definition at line 1192 of file mo_moment.f90.

Referenced by mo_moment::covariance_dp().

Here is the caller graph for this function:

15.16.2.2 real(sp) function mo_moment::covariance::covariance_sp (real(sp), dimension(:), intent(in) *x*, real(sp), dimension(:), intent(in) *y*, logical, dimension(:), intent(in), optional *mask*)

Definition at line 1224 of file mo_moment.f90.

Referenced by mo_moment::covariance_sp().

Here is the caller graph for this function:

The documentation for this interface was generated from the following file:

- mo_moment.f90

15.17 mo_corr::crosscoeffk Interface Reference

Public Member Functions

- real(sp) function crosscoeffk_sp (x, y, k, mask)
- real(dp) function crosscoeffk_dp (x, y, k, mask)

15.17.1 Detailed Description

Definition at line 310 of file mo_corr.f90.

15.17.2 Member Function/Subroutine Documentation

15.17.2.1 real(dp) function mo_corr::crosscoeffk::crosscoeffk_dp (real(dp), dimension(:), intent(in) *x*, real(dp), dimension(:), intent(in) *y*, integer(i4), intent(in) *k*, logical, dimension(:), intent(in), optional *mask*)

Definition at line 916 of file mo_corr.f90.

Referenced by mo_corr::crosscoeffk_dp().

Here is the caller graph for this function:

15.17.2.2 real(sp) function mo_corr::crosscoeffk::crosscoeffk_sp (real(sp), dimension(:), intent(in) *x*, real(sp), dimension(:), intent(in) *y*, integer(i4), intent(in) *k*, logical, dimension(:), intent(in), optional *mask*)

Definition at line 964 of file mo_corr.f90.

Referenced by mo_corr::crosscoeffk_sp().

Here is the caller graph for this function:

The documentation for this interface was generated from the following file:

- mo_corr.f90

15.18 mo_corr::crosscorr Interface Reference

Public Member Functions

- real(sp) function crosscorr_sp (x, y, k, mask)
- real(dp) function crosscorr_dp (x, y, k, mask)

15.18.1 Detailed Description

Definition at line 366 of file mo_corr.f90.

15.18.2 Member Function/Subroutine Documentation

15.18.2.1 real(dp) function mo_corr::crosscorr::crosscorr_dp (real(dp), dimension(:), intent(in) *x*, real(dp), dimension(:), intent(in) *y*, integer(i4), intent(in) *k*, logical, dimension(:), intent(in), optional *mask*)

Definition at line 1014 of file mo_corr.f90.

Referenced by mo_corr::crosscorr_dp().

Here is the caller graph for this function:

15.18.2.2 real(sp) function mo_corr::crosscorr::crosscorr_sp (real(sp), dimension(:), intent(in) *x*, real(sp), dimension(:), intent(in) *y*, integer(i4), intent(in) *k*, logical, dimension(:), intent(in), optional *mask*)

Definition at line 1034 of file mo_corr.f90.

Referenced by mo_corr::crosscorr_sp().

Here is the caller graph for this function:

The documentation for this interface was generated from the following file:

- mo_corr.f90

15.19 mo_orderpack::ctrper Interface Reference

Public Member Functions

- subroutine d_ctrper (XDONT, PCLS)

- subroutine r_ctrper (XDONT, PCLS)
- subroutine i_ctrper (XDONT, PCLS)

15.19.1 Detailed Description

Definition at line 291 of file mo_orderpack.f90.

15.19.2 Member Function/Subroutine Documentation

15.19.2.1 subroutine mo_orderpack::ctrper::d_ctrper (real(kind = dp), dimension (:), intent(inout) *XDONT*, real(kind = dp), intent(in) *PCLS*)

Definition at line 427 of file mo_orderpack.f90.

15.19.2.2 subroutine mo_orderpack::ctrper::i_ctrper (integer(kind = i4), dimension (:), intent(inout) *XDONT*, real(kind = sp), intent(in) *PCLS*)

Definition at line 495 of file mo_orderpack.f90.

15.19.2.3 subroutine mo_orderpack::ctrper::r_ctrper (real(kind = sp), dimension (:), intent(inout) *XDONT*, real(kind = sp), intent(in) *PCLS*)

Definition at line 461 of file mo_orderpack.f90.

The documentation for this interface was generated from the following file:

- mo_orderpack.f90

15.20 mo_temporal_aggregation::day2mon_average Interface Reference

Day-to-month average (day2mon_average)

Public Member Functions

- subroutine day2mon_average_dp (daily_data, yearS, monthS, dayS, mon_avg, misval, rm_misval)

15.20.1 Detailed Description

Day-to-month average (day2mon_average)

converts daily time series to monthly

Parameters

in	*real(sp/dp) :: daily_data(:)*	array of daily time series
in	*integer(i4) :: year*	year of the starting time
in	*integer(i4) :: month*	month of the starting time

in	*integer(i4) :: day*	day of the starting time
in	*real(sp/dp) :: mon_average(:)*	array of monthly averaged values
in	*real(sp/dp) :: misval*	missing value definition
in	*logical :: rm_misval*	switch to exclude missing values

Note

Author

Oldrich Rakovec, Rohini Kumar

Date

Oct 2015

Definition at line 97 of file mo_temporal_aggregation.f90.

15.20.2 Member Function/Subroutine Documentation

15.20.2.1 subroutine mo_temporal_aggregation::day2mon_average::day2mon_average_dp (real(dp), dimension(:), intent(in) *daily_data,* integer(i4), intent(in) *yearS,* integer(i4), intent(in) *monthS,* integer(i4), intent(in) *dayS,* real(dp), dimension(:), intent(inout), allocatable *mon_avg,* real(dp), intent(in), optional *misval,* logical, intent(in), optional *rm_misval*)

Definition at line 166 of file mo_temporal_aggregation.f90.

The documentation for this interface was generated from the following file:

- mo_temporal_aggregation.f90

15.21 mo_ncwrite::dims Type Reference

Collaboration diagram for mo_ncwrite::dims:

Public Attributes

- character(len=maxlen) name
- integer(i4) len
- integer(i4) dimid

15.21.1 Detailed Description

Definition at line 59 of file mo_ncwrite.f90.

15.21.2 Member Data Documentation

15.21.2.1 integer(i4) mo_ncwrite::dims::dimid

Definition at line 62 of file mo_ncwrite.f90.

15.21.2.2 integer(i4) mo_ncwrite::dims::len

Definition at line 61 of file mo_ncwrite.f90.

15.21.2.3 character (len = maxlen) mo_ncwrite::dims::name

Definition at line 60 of file mo_ncwrite.f90.

The documentation for this type was generated from the following file:

- mo_ncwrite.f90

15.22 dummy_mrm Module Reference

15.22.1 Detailed Description

Definition at line 83 of file mrm_driver.f90.

The documentation for this module was generated from the following file:

- mrm_driver.f90

15.23 mo_ncwrite::dump_netcdf Interface Reference

Public Member Functions

- subroutine dump_netcdf_1d_sp (filename, arr, append, lfs, netcdf4, deflate_level)
- subroutine dump_netcdf_2d_sp (filename, arr, append, lfs, netcdf4, deflate_level)
- subroutine dump_netcdf_3d_sp (filename, arr, append, lfs, netcdf4, deflate_level)
- subroutine dump_netcdf_4d_sp (filename, arr, append, lfs, netcdf4, deflate_level)
- subroutine dump_netcdf_5d_sp (filename, arr, append, lfs, netcdf4, deflate_level)
- subroutine dump_netcdf_1d_dp (filename, arr, append, lfs, netcdf4, deflate_level)
- subroutine dump_netcdf_2d_dp (filename, arr, append, lfs, netcdf4, deflate_level)
- subroutine dump_netcdf_3d_dp (filename, arr, append, lfs, netcdf4, deflate_level)
- subroutine dump_netcdf_4d_dp (filename, arr, append, lfs, netcdf4, deflate_level)

- subroutine dump_netcdf_5d_dp (filename, arr, append, lfs, netcdf4, deflate_level)
- subroutine dump_netcdf_1d_i4 (filename, arr, append, lfs, netcdf4, deflate_level)
- subroutine dump_netcdf_2d_i4 (filename, arr, append, lfs, netcdf4, deflate_level)
- subroutine dump_netcdf_3d_i4 (filename, arr, append, lfs, netcdf4, deflate_level)
- subroutine dump_netcdf_4d_i4 (filename, arr, append, lfs, netcdf4, deflate_level)
- subroutine dump_netcdf_5d_i4 (filename, arr, append, lfs, netcdf4, deflate_level)

15.23.1 Detailed Description

Definition at line 162 of file mo_ncwrite.f90.

15.23.2 Member Function/Subroutine Documentation

15.23.2.1 subroutine mo_ncwrite::dump_netcdf::dump_netcdf_1d_dp (character(len = ∗), intent(in) *filename,* real(dp), dimension(:), intent(in) *arr,* logical, intent(in), optional *append,* logical, intent(in), optional *lfs,* logical, intent(in), optional *netcdf4,* integer(i4), intent(in), optional *deflate_level*)

Definition at line 1261 of file mo_ncwrite.f90.

15.23.2.2 subroutine mo_ncwrite::dump_netcdf::dump_netcdf_1d_i4 (character(len = ∗), intent(in) *filename,* integer(i4), dimension(:), intent(in) *arr,* logical, intent(in), optional *append,* logical, intent(in), optional *lfs,* logical, intent(in), optional *netcdf4,* integer(i4), intent(in), optional *deflate_level*)

Definition at line 2009 of file mo_ncwrite.f90.

15.23.2.3 subroutine mo_ncwrite::dump_netcdf::dump_netcdf_1d_sp (character(len = ∗), intent(in) *filename,* real(sp), dimension(:), intent(in) *arr,* logical, intent(in), optional *append,* logical, intent(in), optional *lfs,* logical, intent(in), optional *netcdf4,* integer(i4), intent(in), optional *deflate_level*)

Definition at line 514 of file mo_ncwrite.f90.

15.23.2.4 subroutine mo_ncwrite::dump_netcdf::dump_netcdf_2d_dp (character(len = ∗), intent(in) *filename,* real(dp), dimension(:, :), intent(in) *arr,* logical, intent(in), optional *append,* logical, intent(in), optional *lfs,* logical, intent(in), optional *netcdf4,* integer(i4), intent(in), optional *deflate_level*)

Definition at line 1411 of file mo_ncwrite.f90.

15.23.2.5 subroutine mo_ncwrite::dump_netcdf::dump_netcdf_2d_i4 (character(len = ∗), intent(in) *filename,* integer(i4), dimension(:, :), intent(in) *arr,* logical, intent(in), optional *append,* logical, intent(in), optional *lfs,* logical, intent(in), optional *netcdf4,* integer(i4), intent(in), optional *deflate_level*)

Definition at line 2159 of file mo_ncwrite.f90.

15.23.2.6 subroutine mo_ncwrite::dump_netcdf::dump_netcdf_2d_sp (character(len = ∗), intent(in) *filename,* real(sp), dimension(:, :), intent(in) *arr,* logical, intent(in), optional *append,* logical, intent(in), optional *lfs,* logical, intent(in), optional *netcdf4,* integer(i4), intent(in), optional *deflate_level*)

Definition at line 664 of file mo_ncwrite.f90.

15.23.2.7 subroutine mo_ncwrite::dump_netcdf::dump_netcdf_3d_dp (character(len = ∗), intent(in) *filename,* real(dp), dimension(:, :, :), intent(in) *arr,* logical, intent(in), optional *append,* logical, intent(in), optional *lfs,* logical, intent(in), optional *netcdf4,* integer(i4), intent(in), optional *deflate_level*)

Definition at line 1561 of file mo_ncwrite.f90.

15.23.2.8 subroutine mo_ncwrite::dump_netcdf::dump_netcdf_3d_i4 (character(len = ∗), intent(in) *filename,* integer(i4), dimension(:, :, :), intent(in) *arr,* logical, intent(in), optional *append,* logical, intent(in), optional *lfs,* logical, intent(in), optional *netcdf4,* integer(i4), intent(in), optional *deflate_level*)

Definition at line 2309 of file mo_ncwrite.f90.

15.23.2.9 subroutine mo_ncwrite::dump_netcdf::dump_netcdf_3d_sp (character(len = ∗), intent(in) *filename,* real(sp), dimension(:, :, :), intent(in) *arr,* logical, intent(in), optional *append,* logical, intent(in), optional *lfs,* logical, intent(in), optional *netcdf4,* integer(i4), intent(in), optional *deflate_level*)

Definition at line 814 of file mo_ncwrite.f90.

15.23.2.10 subroutine mo_ncwrite::dump_netcdf::dump_netcdf_4d_dp (character(len = ∗), intent(in) *filename,* real(dp), dimension(:, :, :, :), intent(in) *arr,* logical, intent(in), optional *append,* logical, intent(in), optional *lfs,* logical, intent(in), optional *netcdf4,* integer(i4), intent(in), optional *deflate_level*)

Definition at line 1711 of file mo_ncwrite.f90.

15.23.2.11 subroutine mo_ncwrite::dump_netcdf::dump_netcdf_4d_i4 (character(len = ∗), intent(in) *filename,* integer(i4), dimension(:, :, :, :), intent(in) *arr,* logical, intent(in), optional *append,* logical, intent(in), optional *lfs,* logical, intent(in), optional *netcdf4,* integer(i4), intent(in), optional *deflate_level*)

Definition at line 2458 of file mo_ncwrite.f90.

15.23.2.12 subroutine mo_ncwrite::dump_netcdf::dump_netcdf_4d_sp (character(len = ∗), intent(in) *filename,* real(sp), dimension(:, :, :, :), intent(in) *arr,* logical, intent(in), optional *append,* logical, intent(in), optional *lfs,* logical, intent(in), optional *netcdf4,* integer(i4), intent(in), optional *deflate_level*)

Definition at line 963 of file mo_ncwrite.f90.

15.23.2.13 subroutine mo_ncwrite::dump_netcdf::dump_netcdf_5d_dp (character(len = ∗), intent(in) *filename,* real(dp), dimension(:, :, :, :, :), intent(in) *arr,* logical, intent(in), optional *append,* logical, intent(in), optional *lfs,* logical, intent(in), optional *netcdf4,* integer(i4), intent(in), optional *deflate_level*)

Definition at line 1860 of file mo_ncwrite.f90.

15.23.2.14 subroutine mo_ncwrite::dump_netcdf::dump_netcdf_5d_i4 (character(len = ∗), intent(in) *filename,* integer(i4), dimension(:, :, :, :, :), intent(in) *arr,* logical, intent(in), optional *append,* logical, intent(in), optional *lfs,* logical, intent(in), optional *netcdf4,* integer(i4), intent(in), optional *deflate_level*)

Definition at line 2607 of file mo_ncwrite.f90.

15.23.2.15 subroutine mo_ncwrite::dump_netcdf::dump_netcdf_5d_sp (character(len = ∗), intent(in) *filename,* real(sp), dimension(:, :, :, :, :), intent(in) *arr,* logical, intent(in), optional *append,* logical, intent(in), optional *lfs,* logical, intent(in), optional *netcdf4,* integer(i4), intent(in), optional *deflate_level*)

Definition at line 1112 of file mo_ncwrite.f90.

The documentation for this interface was generated from the following file:

- mo_ncwrite.f90

15.24 mo_utils::eq Interface Reference

Public Member Functions

- elemental pure logical function equal_sp (a, b)
- elemental pure logical function equal_dp (a, b)

15.24.1 Detailed Description

Definition at line 123 of file mo_utils.f90.

15.24.2 Member Function/Subroutine Documentation

15.24.2.1 elemental pure logical function mo_utils::eq::equal_dp (real(dp), intent(in) *a,* real(dp), intent(in) *b*)

Definition at line 419 of file mo_utils.f90.

15.24.2.2 elemental pure logical function mo_utils::eq::equal_sp (real(sp), intent(in) *a,* real(sp), intent(in) *b*)

Definition at line 436 of file mo_utils.f90.

The documentation for this interface was generated from the following file:

- mo_utils.f90

15.25 mo_utils::equal Interface Reference

Comparison of real values.

Public Member Functions

- elemental pure logical function equal_sp (a, b)
- elemental pure logical function equal_dp (a, b)

15.25.1 Detailed Description

Comparison of real values.

Compares two reals if they are numerically equal or not, i.e. equal:

$$|\frac{a-b}{b}| < \varepsilon$$

Parameters

in	*real(sp/dp) :: a*	First number to compare
in	*real(sp/dp) :: b*	Second number to compare

Returns

real(sp/dp) :: equal — $a == b$ logically true or false

Authors

Matthias Cuntz, Juliane Mai

Date

Feb 2014

Definition at line 107 of file mo_utils.f90.

15.25.2 Member Function/Subroutine Documentation

15.25.2.1 elemental pure logical function mo_utils::equal::equal_dp (real(dp), intent(in) *a*, real(dp), intent(in) *b*)

Definition at line 419 of file mo_utils.f90.

Referenced by mo_utils::equal_dp().

Here is the caller graph for this function:

15.25.2.2 elemental pure logical function mo_utils::equal::equal_sp (real(sp), intent(in) *a*, real(sp), intent(in) *b*)

Definition at line 436 of file mo_utils.f90.

Referenced by mo_utils::equal_sp().

Here is the caller graph for this function:

The documentation for this interface was generated from the following file:

- mo_utils.f90

15.26 mo_optimization_utils::eval_interface Interface Reference

Public Member Functions

- subroutine eval_interface (parameterset, runoff, sm_opti, basin_avg_tws, neutrons_opti, et_opti)

15.26.1 Detailed Description

Definition at line 8 of file mo_optimization_utils.f90.

15.26.2 Constructor & Destructor Documentation

15.26.2.1 subroutine mo_optimization_utils::eval_interface::eval_interface (real(dp), dimension(:), intent(in) *parameterset,* real(dp), dimension(:, :), intent(out), optional, allocatable *runoff,* real(dp), dimension(:, :), intent(out), optional, allocatable *sm_opti,* real(dp), dimension(:, :), intent(out), optional, allocatable *basin_avg_tws,* real(dp), dimension(:, :), intent(out), optional, allocatable *neutrons_opti,* real(dp), dimension(:, :), intent(out), optional, allocatable *et_opti*)

Definition at line 8 of file mo_optimization_utils.f90.

The documentation for this interface was generated from the following file:

- mo_optimization_utils.f90

15.27 mo_orderpack::fndnth Interface Reference

Public Member Functions

- real(kind=dp) function d_fndnth (XDONT, NORD)
- real(kind=sp) function r_fndnth (XDONT, NORD)
- integer(kind=i4) function i_fndnth (XDONT, NORD)

15.27.1 Detailed Description

Definition at line 294 of file mo_orderpack.f90.

15.27.2 Member Function/Subroutine Documentation

15.27.2.1 real(kind = dp) function mo_orderpack::fndnth::d_fndnth (real(kind = dp), dimension (:), intent(in) *XDONT,* integer(kind = i4), intent(in) *NORD*)

Definition at line 529 of file mo_orderpack.f90.

15.27.2.2 integer(kind = i4) function mo_orderpack::fndnth::i_fndnth (integer(kind = i4), dimension (:), intent(in) *XDONT,* integer(kind = i4), intent(in) *NORD*)

Definition at line 631 of file mo_orderpack.f90.

15.27.2.3 real(kind = sp) function mo_orderpack::fndnth::r_fndnth (real(kind = sp), dimension (:), intent(in) *XDONT,* integer(kind = i4), intent(in) *NORD*)

Definition at line 580 of file mo_orderpack.f90.

The documentation for this interface was generated from the following file:

- mo_orderpack.f90

15.28 mo_corr::four1 Interface Reference

Private Member Functions

- subroutine four1_sp (data, isign)
- subroutine four1_dp (data, isign)

15.28.1 Detailed Description

Definition at line 380 of file mo_corr.f90.

15.28.2 Member Function/Subroutine Documentation

15.28.2.1 subroutine mo_corr::four1::four1_dp (complex(dpc), dimension(:), intent(inout) *data,* integer(i4), intent(in) *isign*) [private]

Definition at line 1098 of file mo_corr.f90.

15.28.2.2 subroutine mo_corr::four1::four1_sp (complex(spc), dimension(:), intent(inout) *data,* integer(i4), intent(in) *isign*) [private]

Definition at line 1063 of file mo_corr.f90.

The documentation for this interface was generated from the following file:

- mo_corr.f90

15.29 mo_corr::fourrow Interface Reference

Private Member Functions

- subroutine fourrow_sp (data, isign)
- subroutine fourrow_dp (data, isign)

15.29.1 Detailed Description

Definition at line 383 of file mo_corr.f90.

15.29.2 Member Function/Subroutine Documentation

15.29.2.1 subroutine mo_corr::fourrow::fourrow_dp (complex(dpc), dimension(:, :), intent(inout) *data*, integer(i4), intent(in) *isign*) [private]

Definition at line 1190 of file mo_corr.f90.

15.29.2.2 subroutine mo_corr::fourrow::fourrow_sp (complex(spc), dimension(:, :), intent(inout) *data*, integer(i4), intent(in) *isign*) [private]

Definition at line 1140 of file mo_corr.f90.

The documentation for this interface was generated from the following file:

- mo_corr.f90

15.30 mo_mrm_global_variables::gaugingstation Type Reference

Collaboration diagram for mo_mrm_global_variables::gaugingstation:

Public Attributes

- integer(i4), dimension(:), allocatable basinid
- integer(i4), dimension(:), allocatable gaugeid
- character(256), dimension(:), allocatable fname
- real(dp), dimension(:,:), allocatable q

15.30.1 Detailed Description

Definition at line 76 of file mo_mrm_global_variables.f90.

15.30.2 Member Data Documentation

15.30.2.1 integer(i4), dimension(:), allocatable mo_mrm_global_variables::gaugingstation::basinid

Definition at line 77 of file mo_mrm_global_variables.f90.

15.30.2.2 character(256), dimension(:), allocatable mo_mrm_global_variables::gaugingstation::fname

Definition at line 79 of file mo_mrm_global_variables.f90.

15.30.2.3 integer(i4), dimension(:), allocatable mo_mrm_global_variables::gaugingstation::gaugeid

Definition at line 78 of file mo_mrm_global_variables.f90.

15.30.2.4 real(dp), dimension(:, :), allocatable mo_mrm_global_variables::gaugingstation::q

Definition at line 80 of file mo_mrm_global_variables.f90.

The documentation for this type was generated from the following file:

- mo_mrm_global_variables.f90

15.31 mo_utils::ge Interface Reference

Public Member Functions

- elemental pure logical function greaterequal_sp (a, b)
- elemental pure logical function greaterequal_dp (a, b)

15.31.1 Detailed Description

Definition at line 131 of file mo_utils.f90.

15.31.2 Member Function/Subroutine Documentation

15.31.2.1 elemental pure logical function mo_utils::ge::greaterequal_dp (real(dp), intent(in) *a*, real(dp), intent(in) *b*)

Definition at line 454 of file mo_utils.f90.

15.31.2.2 elemental pure logical function mo_utils::ge::greaterequal_sp (real(sp), intent(in) *a*, real(sp), intent(in) *b*)

Definition at line 469 of file mo_utils.f90.

The documentation for this interface was generated from the following file:

- mo_utils.f90

15.32 mo_anneal::generate_neighborhood_weight Interface Reference

Private Member Functions

- subroutine generate_neighborhood_weight_dp (truepara, cum_weight, save_state_xor, iTotalCounter, nIT↩ ERmax, neighborhood)

15.32.1 Detailed Description

Definition at line 349 of file mo_anneal.f90.

15.32.2 Member Function/Subroutine Documentation

15.32.2.1 subroutine mo_anneal::generate_neighborhood_weight::generate_neighborhood_weight_dp (integer(i4), dimension(:), intent(in) *truepara,* real(dp), dimension(:), intent(in) *cum_weight,* integer(i8), dimension(n_save_state), intent(inout) *save_state_xor,* integer(i4), intent(in) *iTotalCounter,* integer(i4), intent(in) *nITERmax,* logical, dimension(size(cum_weight)), intent(out) *neighborhood*) [private]

Definition at line 1465 of file mo_anneal.f90.

Referenced by mo_anneal::anneal_dp().

Here is the caller graph for this function:

The documentation for this interface was generated from the following file:

- mo_anneal.f90

15.33 mo_ncread::get_ncvar Interface Reference

Public Member Functions

- subroutine get_ncvar_0d_sp (Filename, VarName, Dat, fid)
- subroutine get_ncvar_0d_dp (Filename, VarName, Dat, fid)
- subroutine get_ncvar_1d_sp (Filename, VarName, Dat, start, a_count, fid)
- subroutine get_ncvar_1d_dp (Filename, VarName, Dat, start, a_count, fid)
- subroutine get_ncvar_2d_sp (Filename, VarName, Dat, start, a_count, fid)
- subroutine get_ncvar_2d_dp (Filename, VarName, Dat, start, a_count, fid)
- subroutine get_ncvar_3d_sp (Filename, VarName, Dat, start, a_count, fid)
- subroutine get_ncvar_3d_dp (Filename, VarName, Dat, start, a_count, fid)
- subroutine get_ncvar_4d_sp (Filename, VarName, Dat, start, a_count, fid)
- subroutine get_ncvar_4d_dp (Filename, VarName, Dat, start, a_count, fid)
- subroutine get_ncvar_5d_sp (Filename, VarName, Dat, start, a_count, fid)

- subroutine get_ncvar_5d_dp (Filename, VarName, Dat, start, a_count, fid)
- subroutine get_ncvar_0d_i4 (Filename, VarName, Dat, fid)
- subroutine get_ncvar_1d_i4 (Filename, VarName, Dat, start, a_count, fid)
- subroutine get_ncvar_2d_i4 (Filename, VarName, Dat, start, a_count, fid)
- subroutine get_ncvar_3d_i4 (Filename, VarName, Dat, start, a_count, fid)
- subroutine get_ncvar_4d_i4 (Filename, VarName, Dat, start, a_count, fid)
- subroutine get_ncvar_5d_i4 (Filename, VarName, Dat, start, a_count, fid)
- subroutine get_ncvar_0d_i1 (Filename, VarName, Dat, fid)
- subroutine get_ncvar_1d_i1 (Filename, VarName, Dat, start, a_count, fid)
- subroutine get_ncvar_2d_i1 (Filename, VarName, Dat, start, a_count, fid)
- subroutine get_ncvar_3d_i1 (Filename, VarName, Dat, start, a_count, fid)
- subroutine get_ncvar_4d_i1 (Filename, VarName, Dat, start, a_count, fid)
- subroutine get_ncvar_5d_i1 (Filename, VarName, Dat, start, a_count, fid)

15.33.1 Detailed Description

Definition at line 100 of file mo_ncread.f90.

15.33.2 Member Function/Subroutine Documentation

15.33.2.1 subroutine mo_ncread::get_ncvar::get_ncvar_0d_dp (character(len = ∗), intent(in) *Filename,* character(len = ∗), intent(in) *VarName,* real(dp), intent(inout) *Dat,* integer(i4), intent(in), optional *fid*)

Definition at line 437 of file mo_ncread.f90.

15.33.2.2 subroutine mo_ncread::get_ncvar::get_ncvar_0d_i1 (character(len = ∗), intent(in) *Filename,* character(len = ∗), intent(in) *VarName,* integer(1), intent(inout) *Dat,* integer(i4), intent(in), optional *fid*)

Definition at line 1678 of file mo_ncread.f90.

15.33.2.3 subroutine mo_ncread::get_ncvar::get_ncvar_0d_i4 (character(len = ∗), intent(in) *Filename,* character(len = ∗), intent(in) *VarName,* integer(i4), intent(inout) *Dat,* integer(i4), intent(in), optional *fid*)

Definition at line 1259 of file mo_ncread.f90.

15.33.2.4 subroutine mo_ncread::get_ncvar::get_ncvar_0d_sp (character(len = ∗), intent(in) *Filename,* character(len = ∗), intent(in) *VarName,* real(sp), intent(inout) *Dat,* integer(i4), intent(in), optional *fid*)

Definition at line 398 of file mo_ncread.f90.

15.33.2.5 subroutine mo_ncread::get_ncvar::get_ncvar_1d_dp (character(len = ∗), intent(in) *Filename,* character(len = ∗), intent(in) *VarName,* real(dp), dimension(:), intent(inout), allocatable *Dat,* integer(i4), dimension(:), intent(in), optional *start,* integer(i4), dimension(:), intent(in), optional *a_count,* integer(i4), intent(in), optional *fid*)

Definition at line 554 of file mo_ncread.f90.

15.33.2.6 subroutine mo_ncread::get_ncvar::get_ncvar_1d_i1 (character(len = ∗), intent(in) *Filename,* character(len = ∗), intent(in) *VarName,* integer(1), dimension(:), intent(inout), allocatable *Dat,* integer(i4), dimension(:), intent(in), optional *start,* integer(i4), dimension(:), intent(in), optional *a_count,* integer(i4), intent(in), optional *fid*)

Definition at line 1718 of file mo_ncread.f90.

15.33.2.7 subroutine mo_ncread::get_ncvar::get_ncvar_1d_i4 (character(len = ∗), intent(in) *Filename,* character(len = ∗), intent(in) *VarName,* integer(i4), dimension(:), intent(inout), allocatable *Dat,* integer(i4), dimension(:), intent(in), optional *start,* integer(i4), dimension(:), intent(in), optional *a_count,* integer(i4), intent(in), optional *fid*)

Definition at line 1299 of file mo_ncread.f90.

15.33.2.8 subroutine mo_ncread::get_ncvar::get_ncvar_1d_sp (character(len = ∗), intent(in) *Filename,* character(len = ∗), intent(in) *VarName,* real(sp), dimension(:), intent(inout), allocatable *Dat,* integer(i4), dimension(:), intent(in), optional *start,* integer(i4), dimension(:), intent(in), optional *a_count,* integer(i4), intent(in), optional *fid*)

Definition at line 476 of file mo_ncread.f90.

15.33.2.9 subroutine mo_ncread::get_ncvar::get_ncvar_2d_dp (character(len = ∗), intent(in) *Filename,* character(len = ∗), intent(in) *VarName,* real(dp), dimension(:, :), intent(inout), allocatable *Dat,* integer(i4), dimension(:), intent(in), optional *start,* integer(i4), dimension(:), intent(in), optional *a_count,* integer(i4), intent(in), optional *fid*)

Definition at line 711 of file mo_ncread.f90.

15.33.2.10 subroutine mo_ncread::get_ncvar::get_ncvar_2d_i1 (character(len = ∗), intent(in) *Filename,* character(len = ∗), intent(in) *VarName,* integer(1), dimension(:, :), intent(inout), allocatable *Dat,* integer(i4), dimension(:), intent(in), optional *start,* integer(i4), dimension(:), intent(in), optional *a_count,* integer(i4), intent(in), optional *fid*)

Definition at line 1794 of file mo_ncread.f90.

15.33.2.11 subroutine mo_ncread::get_ncvar::get_ncvar_2d_i4 (character(len = ∗), intent(in) *Filename,* character(len = ∗), intent(in) *VarName,* integer(i4), dimension(:, :), intent(inout), allocatable *Dat,* integer(i4), dimension(:), intent(in), optional *start,* integer(i4), dimension(:), intent(in), optional *a_count,* integer(i4), intent(in), optional *fid*)

Definition at line 1375 of file mo_ncread.f90.

15.33.2.12 subroutine mo_ncread::get_ncvar::get_ncvar_2d_sp (character(len = ∗), intent(in) *Filename,* character(len = ∗), intent(in) *VarName,* real(sp), dimension(:, :), intent(inout), allocatable *Dat,* integer(i4), dimension(:), intent(in), optional *start,* integer(i4), dimension(:), intent(in), optional *a_count,* integer(i4), intent(in), optional *fid*)

Definition at line 632 of file mo_ncread.f90.

15.33.2.13 subroutine mo_ncread::get_ncvar::get_ncvar_3d_dp (character(len = ∗), intent(in) *Filename,* character(len = ∗), intent(in) *VarName,* real(dp), dimension(:, :, :), intent(inout), allocatable *Dat,* integer(i4), dimension(:), intent(in), optional *start,* integer(i4), dimension(:), intent(in), optional *a_count,* integer(i4), intent(in), optional *fid*)

Definition at line 869 of file mo_ncread.f90.

15.33.2.14 subroutine mo_ncread::get_ncvar::get_ncvar_3d_i1 (character(len = ∗), intent(in) *Filename,* character(len = ∗), intent(in) *VarName,* integer(1), dimension(:, :, :), intent(inout), allocatable *Dat,* integer(i4), dimension(:), intent(in), optional *start,* integer(i4), dimension(:), intent(in), optional *a_count,* integer(i4), intent(in), optional *fid*)

Definition at line 1870 of file mo_ncread.f90.

15.33.2.15 subroutine mo_ncread::get_ncvar::get_ncvar_3d_i4 (character(len = ∗), intent(in) *Filename,* character(len = ∗), intent(in) *VarName,* integer(i4), dimension(:, :, :), intent(inout), allocatable *Dat,* integer(i4), dimension(:), intent(in), optional *start,* integer(i4), dimension(:), intent(in), optional *a_count,* integer(i4), intent(in), optional *fid*)

Definition at line 1451 of file mo_ncread.f90.

15.33.2.16 subroutine mo_ncread::get_ncvar::get_ncvar_3d_sp (character(len = ∗), intent(in) *Filename,* character(len = ∗), intent(in) *VarName,* real(sp), dimension(:, :, :), intent(inout), allocatable *Dat,* integer(i4), dimension(:), intent(in), optional *start,* integer(i4), dimension(:), intent(in), optional *a_count,* integer(i4), intent(in), optional *fid*)

Definition at line 789 of file mo_ncread.f90.

15.33.2.17 subroutine mo_ncread::get_ncvar::get_ncvar_4d_dp (character(len = ∗), intent(in) *Filename,* character(len = ∗), intent(in) *VarName,* real(dp), dimension(:, :, :, :), intent(inout), allocatable *Dat,* integer(i4), dimension(:), intent(in), optional *start,* integer(i4), dimension(:), intent(in), optional *a_count,* integer(i4), intent(in), optional *fid*)

Definition at line 1025 of file mo_ncread.f90.

15.33.2.18 subroutine mo_ncread::get_ncvar::get_ncvar_4d_i1 (character(len = ∗), intent(in) *Filename,* character(len = ∗), intent(in) *VarName,* integer(1), dimension(:, :, :, :), intent(inout), allocatable *Dat,* integer(i4), dimension(:), intent(in), optional *start,* integer(i4), dimension(:), intent(in), optional *a_count,* integer(i4), intent(in), optional *fid*)

Definition at line 1946 of file mo_ncread.f90.

15.33.2.19 subroutine mo_ncread::get_ncvar::get_ncvar_4d_i4 (character(len = ∗), intent(in) *Filename,* character(len = ∗), intent(in) *VarName,* integer(i4), dimension(:, :, :, :), intent(inout), allocatable *Dat,* integer(i4), dimension(:), intent(in), optional *start,* integer(i4), dimension(:), intent(in), optional *a_count,* integer(i4), intent(in), optional *fid*)

Definition at line 1527 of file mo_ncread.f90.

15.33.2.20 subroutine mo_ncread::get_ncvar::get_ncvar_4d_sp (character(len = ∗), intent(in) *Filename,* character(len = ∗), intent(in) *VarName,* real(sp), dimension(:, :, :, :), intent(inout), allocatable *Dat,* integer(i4), dimension(:), intent(in), optional *start,* integer(i4), dimension(:), intent(in), optional *a_count,* integer(i4), intent(in), optional *fid*)

Definition at line 947 of file mo_ncread.f90.

15.33.2.21 subroutine mo_ncread::get_ncvar::get_ncvar_5d_dp (character(len = ∗), intent(in) *Filename,* character(len = ∗), intent(in) *VarName,* real(dp), dimension(:, :, :, :, :), intent(inout), allocatable *Dat,* integer(i4), dimension(:), intent(in), optional *start,* integer(i4), dimension(:), intent(in), optional *a_count,* integer(i4), intent(in), optional *fid*)

Definition at line 1181 of file mo_ncread.f90.

15.33.2.22 subroutine mo_ncread::get_ncvar::get_ncvar_5d_i1 (character(len = ∗), intent(in) *Filename,* character(len = ∗), intent(in) *VarName,* integer(1), dimension(:, :, :, :, :), intent(inout), allocatable *Dat,* integer(i4), dimension(:), intent(in), optional *start,* integer(i4), dimension(:), intent(in), optional *a_count,* integer(i4), intent(in), optional *fid*)

Definition at line 2022 of file mo_ncread.f90.

15.33.2.23 subroutine mo_ncread::get_ncvar::get_ncvar_5d_i4 (character(len = ∗), intent(in) *Filename,* character(len = ∗), intent(in) *VarName,* integer(i4), dimension(:, :, :, :, :), intent(inout), allocatable *Dat,* integer(i4), dimension(:), intent(in), optional *start,* integer(i4), dimension(:), intent(in), optional *a_count,* integer(i4), intent(in), optional *fid*)

Definition at line 1603 of file mo_ncread.f90.

15.33.2.24 subroutine mo_ncread::get_ncvar::get_ncvar_5d_sp (character(len = ∗), intent(in) *Filename,* character(len = ∗), intent(in) *VarName,* real(sp), dimension(:, :, :, :, :), intent(inout), allocatable *Dat,* integer(i4), dimension(:), intent(in), optional *start,* integer(i4), dimension(:), intent(in), optional *a_count,* integer(i4), intent(in), optional *fid*)

Definition at line 1103 of file mo_ncread.f90.

The documentation for this interface was generated from the following file:

- mo_ncread.f90

15.34 mo_xor4096::get_timeseed Interface Reference

Public Member Functions

- subroutine get_timeseed_i4_0d (seed)
- subroutine get_timeseed_i4_1d (seed)
- subroutine get_timeseed_i8_0d (seed)
- subroutine get_timeseed_i8_1d (seed)

15.34.1 Detailed Description

Definition at line 104 of file mo_xor4096.f90.

15.34.2 Member Function/Subroutine Documentation

15.34.2.1 subroutine mo_xor4096::get_timeseed::get_timeseed_i4_0d (integer(i4), intent(inout) *seed*)

Definition at line 363 of file mo_xor4096.f90.

15.34.2.2 subroutine mo_xor4096::get_timeseed::get_timeseed_i4_1d (integer(i4), dimension(:), intent(inout) *seed*)

Definition at line 380 of file mo_xor4096.f90.

15.34.2.3 subroutine mo_xor4096::get_timeseed::get_timeseed_i8_0d (integer(i8), intent(inout) *seed*)

Definition at line 401 of file mo_xor4096.f90.

15.34.2.4 subroutine mo_xor4096::get_timeseed::get_timeseed_i8_1d (integer(i8), dimension(:), intent(inout) *seed*)

Definition at line 418 of file mo_xor4096.f90.

The documentation for this interface was generated from the following file:

- mo_xor4096.f90

15.35 mo_anneal::gettemperature Interface Reference

GetTemperature.

Public Member Functions

- real(dp) function gettemperature_dp (paraset, cost, eval, acc_goal, prange, prange_func, samplesize, maskpara, seeds, printflag, weight, maxit, undef_funcval)

15.35.1 Detailed Description

GetTemperature.

Determines an initial temperature for Simulated Annealing achieving

Parameters

in	*REAL(DP), DIMENSION(:) :: paraset*	initial (valid) parameter set
in	*INTERFACE :: cost_dp*	interface calculating the cost function at a given point
in	*REAL(DP) :: acc_goal*	Acceptance Ratio which has to be achieved
in	*REAL(DP), DIMENSIO↩ N(size(para),2), optional :: prange*	lower and upper bound per parameter
in	*INTERFACE, optional :: prange_func*	interface calculating the feasible range for a parameter at a certain point, if ranges are variable
in	*INTEGER(I4), optional :: samplesize*	number of iterations the estimation of temperature is based on DEFAULT: Max(20_i4∗n,250_i4)
in	*LOGICAL, DIMENSIO↩ N(size(para)), optional :: maskpara*	maskpara(i) = .true. −> parameter is optimized maskpara(i) = .false. −> parameter is discarded from optimiztaion DEFAULT: .true.
in	*INTEGER(I4/I8), DIMENSION(2), optional :: seeds*	Seeds of random numbers used for random parameter set generation DEFAULT: dependent on current time
in	*LOGICAL, optional :: printflag*	If .true. detailed command line output is written DEFAULT: .false.
in	*REAL(DP), DIMENSIO↩ N(size(para,1)), optional :: weight*	vector of weights per parameter gives the frequency of parameter to be chosen for optimization (will be scaled to a CDF internally) eg. [1,2,1] −> parameter 2 is chosen twice as often as parameter 1 and 2

in	*LOGICAL, optional :: maxit*	minimizing (.false.) or maximizing (.true.) a function DEFAULT: .false. (minimization)
in	*REAL(DP), optional :: undef_funcval*	objective function value defining invalid model output, e.g. -999.0_dp

Returns

real(dp) :: temperature — Temperature achieving a certain acceptance ratio in Simulated Annealing

Note

Either fixed parameter range (prange) OR flexible parameter range (function interface prange_func) has to be given in calling sequence.
Only double precision version available. If single precision is needed not only DP has to be replaced by SP but also I8 of save_state (random number variables) has to be replaced by I4.
ParaChangeMode $>$ 1 is not applied in GetTemperature. For Temperature estimation always only one single parameter is changed (ParaChangeMode=1) which should give theoretically always the best estimate.
Cost and prange_func are user defined functions. See interface definition.

Definition at line 343 of file mo_anneal.f90.

15.35.2 Member Function/Subroutine Documentation

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

Definition at line 1095 of file mo_anneal.f90.

Referenced by mo_anneal::gettemperature_dp().

Here is the caller graph for this function:

The documentation for this interface was generated from the following file:

- mo_anneal.f90

15.36 mo_utils::greaterequal Interface Reference

Public Member Functions

- elemental pure logical function greaterequal_sp (a, b)

- elemental pure logical function greaterequal_dp (a, b)

15.36.1 Detailed Description

Definition at line 115 of file mo_utils.f90.

15.36.2 Member Function/Subroutine Documentation

15.36.2.1 elemental pure logical function mo_utils::greaterequal::greaterequal_dp (real(dp), intent(in) *a*, real(dp), intent(in) *b*)

Definition at line 454 of file mo_utils.f90.

Referenced by mo_utils::greaterequal_dp().

Here is the caller graph for this function:

15.36.2.2 elemental pure logical function mo_utils::greaterequal::greaterequal_sp (real(sp), intent(in) *a*, real(sp), intent(in) *b*)

Definition at line 469 of file mo_utils.f90.

Referenced by mo_utils::greaterequal_sp().

Here is the caller graph for this function:

The documentation for this interface was generated from the following file:

- mo_utils.f90

15.37 mo_common_variables::grid Type Reference

Collaboration diagram for mo_common_variables::grid:

Public Attributes

- integer(i4) ncols
- integer(i4) nrows
- integer(i4) ncells
- real(dp) xllcorner
- real(dp) yllcorner
- real(dp) cellsize
- real(dp) nodata_value
- real(dp), dimension(:,:), allocatable x
- real(dp), dimension(:,:), allocatable y
- logical, dimension(:,:), allocatable mask
- integer(i4) istart
- integer(i4) iend
- integer(i4), dimension(:,:), allocatable cellcoor
- real(dp), dimension(:), allocatable cellarea
- integer(i4), dimension(:), allocatable id

15.37.1 Detailed Description

Definition at line 73 of file mo_common_variables.f90.

15.37.2 Member Data Documentation

15.37.2.1 real(dp), dimension(:), allocatable mo_common_variables::grid::cellarea

Definition at line 90 of file mo_common_variables.f90.

15.37.2.2 integer(i4), dimension(:, :), allocatable mo_common_variables::grid::cellcoor

Definition at line 89 of file mo_common_variables.f90.

15.37.2.3 real(dp) mo_common_variables::grid::cellsize

Definition at line 80 of file mo_common_variables.f90.

15.37.2.4 integer(i4), dimension(:), allocatable mo_common_variables::grid::id

Definition at line 91 of file mo_common_variables.f90.

15.37.2.5 integer(i4) mo_common_variables::grid::iend

Definition at line 87 of file mo_common_variables.f90.

15.37.2.6 integer(i4) mo_common_variables::grid::istart

Definition at line 86 of file mo_common_variables.f90.

15.37.2.7 logical, dimension(:, :), allocatable mo_common_variables::grid::mask

Definition at line 84 of file mo_common_variables.f90.

15.37.2.8 integer(i4) mo_common_variables::grid::ncells

Definition at line 77 of file mo_common_variables.f90.

15.37.2.9 integer(i4) mo_common_variables::grid::ncols

Definition at line 75 of file mo_common_variables.f90.

15.37.2.10 real(dp) mo_common_variables::grid::nodata_value

Definition at line 81 of file mo_common_variables.f90.

15.37.2.11 integer(i4) mo_common_variables::grid::nrows

Definition at line 76 of file mo_common_variables.f90.

15.37.2.12 real(dp), dimension(:, :), allocatable mo_common_variables::grid::x

Definition at line 82 of file mo_common_variables.f90.

15.37.2.13 real(dp) mo_common_variables::grid::xllcorner

Definition at line 78 of file mo_common_variables.f90.

15.37.2.14 real(dp), dimension(:, :), allocatable mo_common_variables::grid::y

Definition at line 83 of file mo_common_variables.f90.

15.37.2.15 real(dp) mo_common_variables::grid::yllcorner

Definition at line 79 of file mo_common_variables.f90.

The documentation for this type was generated from the following file:

- mo_common_variables.f90

15.38 mo_common_variables::gridremapper Type Reference

Collaboration diagram for mo_common_variables::gridremapper:

Public Attributes

- type(grid), pointer high_res_grid
- type(grid), pointer low_res_grid
- integer(i4), dimension(:), allocatable lower_bound
- integer(i4), dimension(:), allocatable upper_bound
- integer(i4), dimension(:), allocatable left_bound
- integer(i4), dimension(:), allocatable right_bound
- integer(i4), dimension(:), allocatable n_subcells
- integer(i4), dimension(:,:), allocatable lowres_id_on_highres

15.38.1 Detailed Description

Definition at line 98 of file mo_common_variables.f90.

15.38.2 Member Data Documentation

15.38.2.1 type(grid), pointer mo_common_variables::gridremapper::high_res_grid

Definition at line 99 of file mo_common_variables.f90.

15.38.2.2 integer(i4), dimension(:), allocatable mo_common_variables::gridremapper::left_bound

Definition at line 105 of file mo_common_variables.f90.

15.38.2.3 type(grid), pointer mo_common_variables::gridremapper::low_res_grid

Definition at line 100 of file mo_common_variables.f90.

15.38.2.4 integer(i4), dimension(:), allocatable mo_common_variables::gridremapper::lower_bound

Definition at line 103 of file mo_common_variables.f90.

15.38.2.5 integer(i4), dimension(:, :), allocatable mo_common_variables::gridremapper::lowres_id_on_highres

Definition at line 108 of file mo_common_variables.f90.

15.38.2.6 integer(i4), dimension(:), allocatable mo_common_variables::gridremapper::n_subcells

Definition at line 107 of file mo_common_variables.f90.

15.38.2.7 integer(i4), dimension(:), allocatable mo_common_variables::gridremapper::right_bound

Definition at line 106 of file mo_common_variables.f90.

15.38.2.8 integer(i4), dimension(:), allocatable mo_common_variables::gridremapper::upper_bound

Definition at line 104 of file mo_common_variables.f90.

The documentation for this type was generated from the following file:

- mo_common_variables.f90

15.39 mo_temporal_aggregation::hour2day_average Interface Reference

Hour-to-day average (hour2day_average)

Public Member Functions

- subroutine hour2day_average_dp (hourly_data, yearS, monthS, dayS, hourS, day_avg, misval, rm_misval)

15.39.1 Detailed Description

Hour-to-day average (hour2day_average)

converts hourly time series to daily

Parameters

in	*real(sp/dp) :: hourly_data(:)*	array of hourly time series
in	*integer(i4) :: year*	year of the starting time
in	*integer(i4) :: month*	month of the starting time
in	*integer(i4) :: day*	day of the starting time
in	*integer(i4) :: hour*	hour of the starting time
in	*real(sp/dp) :: day_average(:)*	array of daily averaged values
in	*real(sp/dp) :: misval*	missing value definition
in	*logical :: rm_misval*	switch to exclude missing values

Note

Hours values should be from 0 to 23 (NOT from 1 to 24!)

Author

Oldrich Rakovec, Rohini Kumar

Date

Oct 2015

Definition at line 154 of file mo_temporal_aggregation.f90.

15.39.2 Member Function/Subroutine Documentation

15.39.2.1 subroutine mo_temporal_aggregation::hour2day_average::hour2day_average_dp (real(dp), dimension(:), intent(in) *hourly_data,* integer(i4), intent(in) *yearS,* integer(i4), intent(in) *monthS,* integer(i4), intent(in) *dayS,* integer(i4), intent(in) *hourS,* real(dp), dimension(:), intent(inout), allocatable *day_avg,* real(dp), intent(in), optional *misval,* logical, intent(in), optional *rm_misval*)

Definition at line 273 of file mo_temporal_aggregation.f90.

The documentation for this interface was generated from the following file:

- mo_temporal_aggregation.f90

15.40 mo_orderpack::indmed Interface Reference

Public Member Functions

- subroutine d_indmed (XDONT, INDM)
- subroutine r_indmed (XDONT, INDM)
- subroutine i_indmed (XDONT, INDM)

15.40.1 Detailed Description

Definition at line 297 of file mo_orderpack.f90.

15.40.2 Member Function/Subroutine Documentation

15.40.2.1 subroutine mo_orderpack::indmed::d_indmed (real(kind = dp), dimension (:), intent(in) *XDONT,* integer(kind = i4), intent(out) *INDM*)

Definition at line 682 of file mo_orderpack.f90.

15.40.2.2 subroutine mo_orderpack::indmed::i_indmed (integer(kind = i4), dimension (:), intent(in) *XDONT,* integer(kind = i4), intent(out) *INDM*)

Definition at line 1478 of file mo_orderpack.f90.

15.40.2.3 subroutine mo_orderpack::indmed::r_indmed (real(kind = sp), dimension (:), intent(in) *XDONT,* integer(kind = i4), intent(out) *INDM*)

Definition at line 1080 of file mo_orderpack.f90.

The documentation for this interface was generated from the following file:

- mo_orderpack.f90

15.41 mo_orderpack::indnth Interface Reference

Public Member Functions

- integer(kind=i4) function d_indnth (XDONT, NORD)
- integer(kind=i4) function r_indnth (XDONT, NORD)
- integer(kind=i4) function i_indnth (XDONT, NORD)

15.41.1 Detailed Description

Definition at line 300 of file mo_orderpack.f90.

15.41.2 Member Function/Subroutine Documentation

15.41.2.1 integer(kind = i4) function mo_orderpack::indnth::d_indnth (real(kind = dp), dimension (:), intent(in) *XDONT,* integer(kind = i4), intent(in) *NORD*)

Definition at line 1876 of file mo_orderpack.f90.

15.41.2.2 integer(kind = i4) function mo_orderpack::indnth::i_indnth (integer(kind = i4), dimension (:), intent(in) *XDONT,* integer(kind = i4), intent(in) *NORD*)

Definition at line 2888 of file mo_orderpack.f90.

15.41.2.3 integer(kind = i4) function mo_orderpack::indnth::r_indnth (real(kind = sp), dimension (:), intent(in) *XDONT,* integer(kind = i4), intent(in) *NORD*)

Definition at line 2382 of file mo_orderpack.f90.

The documentation for this interface was generated from the following file:

- mo_orderpack.f90

15.42 mo_orderpack::inspar Interface Reference

Public Member Functions

- subroutine d_inspar (XDONT, NORD)
- subroutine r_inspar (XDONT, NORD)
- subroutine i_inspar (XDONT, NORD)

15.42.1 Detailed Description

Definition at line 303 of file mo_orderpack.f90.

15.42.2 Member Function/Subroutine Documentation

15.42.2.1 subroutine mo_orderpack::inspar::d_inspar (real(kind = dp), dimension (:), intent(inout) *XDONT,* integer(kind = i4), intent(in) *NORD*)

Definition at line 3394 of file mo_orderpack.f90.

15.42.2.2 subroutine mo_orderpack::inspar::i_inspar (integer(kind = i4), dimension (:), intent(inout) *XDONT,* integer(kind = i4), intent(in) *NORD*)

Definition at line 3484 of file mo_orderpack.f90.

15.42.2.3 subroutine mo_orderpack::inspar::r_inspar (real(kind = sp), dimension (:), intent(inout) *XDONT*, integer(kind = i4), intent(in) *NORD*)

Definition at line 3439 of file mo_orderpack.f90.

The documentation for this interface was generated from the following file:

- mo_orderpack.f90

15.43 mo_orderpack::inssor Interface Reference

Public Member Functions

- subroutine d_inssor (XDONT)
- subroutine r_inssor (XDONT)
- subroutine i_inssor (XDONT)

15.43.1 Detailed Description

Definition at line 306 of file mo_orderpack.f90.

15.43.2 Member Function/Subroutine Documentation

15.43.2.1 subroutine mo_orderpack::inssor::d_inssor (real(kind = dp), dimension (:), intent(inout) *XDONT*)

Definition at line 3529 of file mo_orderpack.f90.

Referenced by mo_orderpack::d_refsor().

Here is the caller graph for this function:

15.43.2.2 subroutine mo_orderpack::inssor::i_inssor (integer(kind = i4), dimension (:), intent(inout) *XDONT*)

Definition at line 3657 of file mo_orderpack.f90.

Referenced by mo_orderpack::i_refsor().

Here is the caller graph for this function:

15.43.2.3 subroutine mo_orderpack::inssor::r_inssor (real(kind = sp), dimension (:), intent(inout) *XDONT*)

Definition at line 3593 of file mo_orderpack.f90.

Referenced by mo_orderpack::r_refsor().

Here is the caller graph for this function:

The documentation for this interface was generated from the following file:

- mo_orderpack.f90

15.44 mo_utils::is_finite Interface Reference

.true. if not IEEE Inf, IEEE NaN, nor IEEE Inf nor IEEE NaN, respectively.

Public Member Functions

- elemental pure logical function is_finite_sp (a)
- elemental pure logical function is_finite_dp (a)

15.44.1 Detailed Description

.true. if not IEEE Inf, IEEE NaN, nor IEEE Inf nor IEEE NaN, respectively.

Checks for IEEE Inf and IEEE NaN, i.e. Infinity and Not-a-Number.
Wraps to functions of the intrinsic module ieee_arithmetic but gives alternatives for gfortran, which does not provide ieee_arithmetic.

Parameters

in	*real(sp/dp) :: x*	Number to check

Returns

logical :: is_finite/is_nan/is_normal — $a/=Inf, a==NaN, a/=Inf and a==NaN$, logically true or false

Authors

Matthias Cuntz

Date

Mar 2015

Definition at line 193 of file mo_utils.f90.

15.44.2 Member Function/Subroutine Documentation

15.44.2.1 elemental pure logical function mo_utils::is_finite::is_finite_dp (real(dp), intent(in) *a*)

Definition at line 551 of file mo_utils.f90.

Referenced by mo_utils::is_finite_dp().

Here is the caller graph for this function:

15.44.2.2 elemental pure logical function mo_utils::is_finite::is_finite_sp (real(sp), intent(in) *a*)

Definition at line 570 of file mo_utils.f90.

Referenced by mo_utils::is_finite_sp().

Here is the caller graph for this function:

The documentation for this interface was generated from the following file:

- mo_utils.f90

15.45 mo_utils::is_nan Interface Reference

Public Member Functions

- elemental pure logical function is_nan_sp (a)
- elemental pure logical function is_nan_dp (a)

15.45.1 Detailed Description

Definition at line 197 of file mo_utils.f90.

15.45.2 Member Function/Subroutine Documentation

15.45.2.1 elemental pure logical function mo_utils::is_nan::is_nan_dp (real(dp), intent(in) *a*)

Definition at line 590 of file mo_utils.f90.

Referenced by mo_utils::is_nan_dp().

Here is the caller graph for this function:

15.45.2.2 elemental pure logical function mo_utils::is_nan::is_nan_sp (real(sp), intent(in) *a*)

Definition at line 610 of file mo_utils.f90.

Referenced by mo_utils::is_nan_sp().

Here is the caller graph for this function:

The documentation for this interface was generated from the following file:

- mo_utils.f90

15.46 mo_utils::is_normal Interface Reference

Public Member Functions

- elemental pure logical function is_normal_sp (a)
- elemental pure logical function is_normal_dp (a)

15.46.1 Detailed Description

Definition at line 201 of file mo_utils.f90.

15.46.2 Member Function/Subroutine Documentation

15.46.2.1 elemental pure logical function mo_utils::is_normal::is_normal_dp (real(dp), intent(in) *a*)

Definition at line 631 of file mo_utils.f90.

Referenced by mo_utils::is_normal_dp().

Here is the caller graph for this function:

15.46.2.2 elemental pure logical function mo_utils::is_normal::is_normal_sp (real(sp), intent(in) *a*)

Definition at line 650 of file mo_utils.f90.

Referenced by mo_utils::is_normal_sp().

Here is the caller graph for this function:

The documentation for this interface was generated from the following file:

- mo_utils.f90

15.47 mo_errormeasures::kge Interface Reference

Kling-Gupta-Efficiency measure.

Public Member Functions

- real(dp) function kge_dp_1d (x, y, mask)
- real(dp) function kge_dp_2d (x, y, mask)
- real(dp) function kge_dp_3d (x, y, mask)
- real(sp) function kge_sp_1d (x, y, mask)
- real(sp) function kge_sp_2d (x, y, mask)
- real(sp) function kge_sp_3d (x, y, mask)

15.47.1 Detailed Description

Kling-Gupta-Efficiency measure.

The Kling-Gupta model efficiency coefficient KGE is

$$KGE = 1 - \sqrt{((1-r)^2 + (1-\alpha)^2 + (1-\beta)^2)}$$

where
r = Pearson product-moment correlation coefficient
α = ratio of simulated mean to observed mean
β = ratio of simulated standard deviation to observed standard deviation
This three measures are calculated between two arrays (1d, 2d, or 3d). Usually, one is an observation and the second is a modelled variable.
The higher the KGE the better the observation and simulation are matching. The upper limit of KGE is 1.
Therefore, if you apply a minimization algorithm to calibrate regarding KGE you have to use the objective function

$$obj_value = 1.0 - KGE$$

which has then the optimum at 0.0. (Like for the NSE where you always optimize 1-NSE.)
real(sp/dp), dimension(:) :: x, y 1D-array with input numbers real(sp/dp), dimension(:,:) :: x, y 2D-array with input numbers real(sp/dp), dimension(:,:,:) :: x, y 3D-array with input numbers logical :: mask(:) 1D-array of logical values with size(x/y). logical :: mask(:,:) 2D-array of logical values with size(x/y). logical :: mask(:,:,:) 3D-array of logical values with size(x/y).

Returns

kge — Kling-Gupta-Efficiency (value less equal 1.0)

Note

Input values must be floating points.
Gupta, Hoshin V., et al. "Decomposition of the mean squared error and NSE performance criteria: Implications for improving hydrological modelling." Journal of Hydrology 377.1 (2009): 80-91.

Author

Rohini Kumar

Date

August 2014

Definition at line 186 of file mo_errormeasures.f90.

15.47.2 Member Function/Subroutine Documentation

15.47.2.1 real(dp) function mo_errormeasures::kge::kge_dp_1d (real(dp), dimension(:), intent(in) *x*, real(dp), dimension(:), intent(in) *y*, logical, dimension(:), intent(in), optional *mask*)

Definition at line 1203 of file mo_errormeasures.f90.

Referenced by mo_errormeasures::kge_dp_1d().

Here is the caller graph for this function:

15.47.2.2 real(dp) function mo_errormeasures::kge::kge_dp_2d (real(dp), dimension(:, :), intent(in) *x*, real(dp), dimension(:, :), intent(in) *y*, logical, dimension(:, :), intent(in), optional *mask*)

Definition at line 1256 of file mo_errormeasures.f90.

Referenced by mo_errormeasures::kge_dp_2d().

Here is the caller graph for this function:

15.47.2.3 real(dp) function mo_errormeasures::kge::kge_dp_3d (real(dp), dimension(:, :, :), intent(in) *x*, real(dp), dimension(:, :, :), intent(in) *y*, logical, dimension(:, :, :), intent(in), optional *mask*)

Definition at line 1320 of file mo_errormeasures.f90.

Referenced by mo_errormeasures::kge_dp_3d().

Here is the caller graph for this function:

15.47.2.4 real(sp) function mo_errormeasures::kge::kge_sp_1d (real(sp), dimension(:), intent(in) *x*, real(sp), dimension(:), intent(in) *y*, logical, dimension(:), intent(in), optional *mask*)

Definition at line 1022 of file mo_errormeasures.f90.

Referenced by mo_errormeasures::kge_sp_1d().

Here is the caller graph for this function:

15.47.2.5 real(sp) function mo_errormeasures::kge::kge_sp_2d (real(sp), dimension(:, :), intent(in) *x*, real(sp), dimension(:, :), intent(in) *y*, logical, dimension(:, :), intent(in), optional *mask*)

Definition at line 1075 of file mo_errormeasures.f90.

Referenced by mo_errormeasures::kge_sp_2d().

Here is the caller graph for this function:

15.47.2.6 real(sp) function mo_errormeasures::kge::kge_sp_3d (real(sp), dimension(:, :, :), intent(in) *x*, real(sp), dimension(:, :, :), intent(in) *y*, logical, dimension(:, :, :), intent(in), optional *mask*)

Definition at line 1139 of file mo_errormeasures.f90.

Referenced by mo_errormeasures::kge_sp_3d().

Here is the caller graph for this function:

The documentation for this interface was generated from the following file:

- mo_errormeasures.f90

15.48 mo_errormeasures::kgenocorr Interface Reference

Kling-Gupta-Efficiency measure without correlation.

Public Member Functions

- real(dp) function kgenocorr_dp_1d (x, y, mask)
- real(dp) function kgenocorr_dp_2d (x, y, mask)
- real(dp) function kgenocorr_dp_3d (x, y, mask)
- real(sp) function kgenocorr_sp_1d (x, y, mask)
- real(sp) function kgenocorr_sp_2d (x, y, mask)
- real(sp) function kgenocorr_sp_3d (x, y, mask)

15.48.1 Detailed Description

Kling-Gupta-Efficiency measure without correlation.

The modified Kling-Gupta model efficiency coefficient $KGEnocorr$ is

$$KGEnocorr = 1 - \sqrt{((1-\alpha)^2 + (1-\beta)^2)}$$

where
α = ratio of simulated mean to observed mean
β = ratio of simulated standard deviation to observed standard deviation
This two measures are calculated between two arrays (1d, 2d, or 3d). Usually, one is an observation and the second is a modelled variable.
The higher the KGEnocorr the better the observation and simulation are matching. The upper limit of KGEnocorr is 1.
Therefore, if you apply a minimization algorithm to calibrate regarding KGEnocorr you have to use the objective function

$$obj_value = 1.0 - KGEnocorr$$

which has then the optimum at 0.0. (Like for the NSE where you always optimize 1-NSE.)
real(sp/dp), dimension(:) :: x, y 1D-array with input numbers real(sp/dp), dimension(:,:) :: x, y 2D-array with input numbers real(sp/dp), dimension(:,:,:) :: x, y 3D-array with input numbers logical :: mask(:) 1D-array of logical values with size(x/y). logical :: mask(:,:) 2D-array of logical values with size(x/y). logical :: mask(:,:,:) 3D-array of logical values with size(x/y).

Returns

kgenocorr — Kling-Gupta-Efficiency without correlation (value less equal 1.0)

Note

Input values must be floating points.
Gupta, Hoshin V., et al. "Decomposition of the mean squared error and NSE performance criteria: Implications for improving hydrological modelling." Journal of Hydrology 377.1 (2009): 80-91.

Author

Rohini Kumar

Date

August 2014

Definition at line 268 of file mo_errormeasures.f90.

15.48.2 Member Function/Subroutine Documentation

15.48.2.1 real(dp) function mo_errormeasures::kgenocorr::kgenocorr_dp_1d (real(dp), dimension(:), intent(in) *x*, real(dp), dimension(:), intent(in) *y*, logical, dimension(:), intent(in), optional *mask*)

Definition at line 1549 of file mo_errormeasures.f90.

Referenced by mo_errormeasures::kgenocorr_dp_1d().

Here is the caller graph for this function:

15.48.2.2 real(dp) function mo_errormeasures::kgenocorr::kgenocorr_dp_2d (real(dp), dimension(:, :), intent(in) *x*, real(dp), dimension(:, :), intent(in) *y*, logical, dimension(:, :), intent(in), optional *mask*)

Definition at line 1599 of file mo_errormeasures.f90.

Referenced by mo_errormeasures::kgenocorr_dp_2d().

Here is the caller graph for this function:

15.48.2.3 real(dp) function mo_errormeasures::kgenocorr::kgenocorr_dp_3d (real(dp), dimension(:, :, :), intent(in) *x*, real(dp), dimension(:, :, :), intent(in) *y*, logical, dimension(:, :, :), intent(in), optional *mask*)

Definition at line 1655 of file mo_errormeasures.f90.

Referenced by mo_errormeasures::kgenocorr_dp_3d().

Here is the caller graph for this function:

15.48.2.4 real(sp) function mo_errormeasures::kgenocorr::kgenocorr_sp_1d (real(sp), dimension(:), intent(in) *x*, real(sp), dimension(:), intent(in) *y*, logical, dimension(:), intent(in), optional *mask*)

Definition at line 1386 of file mo_errormeasures.f90.

Referenced by mo_errormeasures::kgenocorr_sp_1d().

Here is the caller graph for this function:

15.48.2.5 real(sp) function mo_errormeasures::kgenocorr::kgenocorr_sp_2d (real(sp), dimension(:, :), intent(in) *x*, real(sp), dimension(:, :), intent(in) *y*, logical, dimension(:, :), intent(in), optional *mask*)

Definition at line 1436 of file mo_errormeasures.f90.

Referenced by mo_errormeasures::kgenocorr_sp_2d().

Here is the caller graph for this function:

15.48.2.6 real(sp) function mo_errormeasures::kgenocorr::kgenocorr_sp_3d (real(sp), dimension(:, :, :), intent(in) *x*, real(sp), dimension(:, :, :), intent(in) *y*, logical, dimension(:, :, :), intent(in), optional *mask*)

Definition at line 1492 of file mo_errormeasures.f90.

Referenced by mo_errormeasures::kgenocorr_sp_3d().

Here is the caller graph for this function:

The documentation for this interface was generated from the following file:

- mo_errormeasures.f90

15.49 mo_moment::kurtosis Interface Reference

Public Member Functions

- real(sp) function kurtosis_sp (dat, mask)
- real(dp) function kurtosis_dp (dat, mask)

15.49.1 Detailed Description

Definition at line 437 of file mo_moment.f90.

15.49.2 Member Function/Subroutine Documentation

15.49.2.1 real(dp) function mo_moment::kurtosis::kurtosis_dp (real(dp), dimension(:), intent(in) *dat*, logical, dimension(:), intent(in), optional *mask*)

Definition at line 1257 of file mo_moment.f90.

Referenced by mo_moment::kurtosis_dp().

Here is the caller graph for this function:

15.49.2.2 real(sp) function mo_moment::kurtosis::kurtosis_sp (real(sp), dimension(:), intent(in) *dat,* logical, dimension(:), intent(in), optional *mask*)

Definition at line 1298 of file mo_moment.f90.

Referenced by mo_moment::kurtosis_sp().

Here is the caller graph for this function:

The documentation for this interface was generated from the following file:

- mo_moment.f90

15.50 mo_utils::le Interface Reference

Public Member Functions

- elemental pure logical function lesserequal_sp (a, b)
- elemental pure logical function lesserequal_dp (a, b)

15.50.1 Detailed Description

Definition at line 135 of file mo_utils.f90.

15.50.2 Member Function/Subroutine Documentation

15.50.2.1 elemental pure logical function mo_utils::le::lesserequal_dp (real(dp), intent(in) *a*, real(dp), intent(in) *b*)

Definition at line 485 of file mo_utils.f90.

15.50.2.2 elemental pure logical function mo_utils::le::lesserequal_sp (real(sp), intent(in) *a*, real(sp), intent(in) *b*)

Definition at line 500 of file mo_utils.f90.

The documentation for this interface was generated from the following file:

- mo_utils.f90

15.51 mo_utils::lesserequal Interface Reference

Public Member Functions

- elemental pure logical function lesserequal_sp (a, b)
- elemental pure logical function lesserequal_dp (a, b)

15.51.1 Detailed Description

Definition at line 119 of file mo_utils.f90.

15.51.2 Member Function/Subroutine Documentation

15.51.2.1 elemental pure logical function mo_utils::lesserequal::lesserequal_dp (real(dp), intent(in) *a*, real(dp), intent(in) *b*)

Definition at line 485 of file mo_utils.f90.

Referenced by mo_utils::lesserequal_dp().

Here is the caller graph for this function:

15.51.2.2 elemental pure logical function mo_utils::lesserequal::lesserequal_sp (real(sp), intent(in) *a*, real(sp), intent(in) *b*)

Definition at line 500 of file mo_utils.f90.

Referenced by mo_utils::lesserequal_sp().

Here is the caller graph for this function:

The documentation for this interface was generated from the following file:

- mo_utils.f90

15.52 mo_linfit::linfit Interface Reference

Fits a straight line to input data by minimizing chi^2.

Public Member Functions

- real(sp) function, dimension(:),
  allocatable linfit_sp (x, y, a, b, siga, sigb, chi2, model2)
- real(dp) function, dimension(:),
  allocatable linfit_dp (x, y, a, b, siga, sigb, chi2, model2)

15.52.1 Detailed Description

Fits a straight line to input data by minimizing chi^2.

Given a set of data points x(1:ndata), y(1:ndata), fit them to a straight line $y = a + bx$ by minimizing chi2.
Model I minimizes y vs. x while Model II takes the geometric mean of y vs. x and x vs. y. Returned is the fitted line at x. Optional returns are a, b and their respective probable uncertainties siga and sigb, and the chi-square chi2.

Parameters

in	*real(sp/dp) :: x(:)*	1D-array with input x
in	*real(sp/dp) :: y(:)*	1D-array with input y
in	*logical, optional :: model2*	If present, use geometric mean regression instead of ordinary least square
out	*real(sp/dp), dimension(M) :: a*	intercept
out	*real(sp/dp), dimension(M) :: b*	slope

out	*real(sp/dp), dimension(M) :: siga*	error on intercept
out	*real(sp/dp), dimension(M) :: sigb*	error on slope
out	*real(sp/dp) :: chisq*	Minimum chi^2

Returns

real(sp/dp), dimension(:), allocatable :: out — fitted values at x(:).

Definition at line 106 of file mo_linfit.f90.

15.52.2 Member Function/Subroutine Documentation

15.52.2.1 real(dp) function, dimension(:), allocatable mo_linfit::linfit::linfit_dp (real(dp), dimension(:), intent(in) *x,* real(dp), dimension(:), intent(in) *y,* real(dp), intent(out), optional *a,* real(dp), intent(out), optional *b,* real(dp), intent(out), optional *siga,* real(dp), intent(out), optional *sigb,* real(dp), intent(out), optional *chi2,* logical, intent(in), optional *model2*)

Definition at line 120 of file mo_linfit.f90.

Referenced by mo_linfit::linfit_dp().

Here is the caller graph for this function:

15.52.2.2 real(sp) function, dimension(:), allocatable mo_linfit::linfit::linfit_sp (real(sp), dimension(:), intent(in) *x,* real(sp), dimension(:), intent(in) *y,* real(sp), intent(out), optional *a,* real(sp), intent(out), optional *b,* real(sp), intent(out), optional *siga,* real(sp), intent(out), optional *sigb,* real(sp), intent(out), optional *chi2,* logical, intent(in), optional *model2*)

Definition at line 211 of file mo_linfit.f90.

Referenced by mo_linfit::linfit_sp().

Here is the caller graph for this function:

The documentation for this interface was generated from the following file:

- mo_linfit.f90

15.53 mo_errormeasures::lnnse Interface Reference

Public Member Functions

- real(sp) function lnnse_sp_1d (x, y, mask)
- real(dp) function lnnse_dp_1d (x, y, mask)
- real(dp) function lnnse_dp_2d (x, y, mask)
- real(sp) function lnnse_sp_2d (x, y, mask)
- real(sp) function lnnse_sp_3d (x, y, mask)
- real(dp) function lnnse_dp_3d (x, y, mask)

15.53.1 Detailed Description

Definition at line 334 of file mo_errormeasures.f90.

15.53.2 Member Function/Subroutine Documentation

15.53.2.1 real(dp) function mo_errormeasures::lnnse::lnnse_dp_1d (real(dp), dimension(:), intent(in) *x*, real(dp), dimension(:), intent(in) *y*, logical, dimension(:), intent(inout), optional *mask*)

Definition at line 1772 of file mo_errormeasures.f90.

Referenced by mo_errormeasures::lnnse_dp_1d().

Here is the caller graph for this function:

15.53.2.2 real(dp) function mo_errormeasures::lnnse::lnnse_dp_2d (real(dp), dimension(:, :), intent(in) *x*, real(dp), dimension(:, :), intent(in) *y*, logical, dimension(:, :), intent(inout), optional *mask*)

Definition at line 1886 of file mo_errormeasures.f90.

Referenced by mo_errormeasures::lnnse_dp_2d().

Here is the caller graph for this function:

15.53.2.3 real(dp) function mo_errormeasures::lnnse::lnnse_dp_3d (real(dp), dimension(:, :, :), intent(in) *x*, real(dp), dimension(:, :, :), intent(in) *y*, logical, dimension(:, :, :), intent(inout), optional *mask*)

Definition at line 2000 of file mo_errormeasures.f90.

Referenced by mo_errormeasures::lnnse_dp_3d().

Here is the caller graph for this function:

15.53.2.4 real(sp) function mo_errormeasures::lnnse::lnnse_sp_1d (real(sp), dimension(:), intent(in) *x*, real(sp), dimension(:), intent(in) *y*, logical, dimension(:), intent(inout), optional *mask*)

Definition at line 1715 of file mo_errormeasures.f90.

Referenced by mo_errormeasures::lnnse_sp_1d().

Here is the caller graph for this function:

15.53.2.5 real(sp) function mo_errormeasures::lnnse::lnnse_sp_2d (real(sp), dimension(:, :), intent(in) *x*, real(sp), dimension(:, :), intent(in) *y*, logical, dimension(:, :), intent(inout), optional *mask*)

Definition at line 1829 of file mo_errormeasures.f90.

Referenced by mo_errormeasures::lnnse_sp_2d().

Here is the caller graph for this function:

15.53.2.6 real(sp) function mo_errormeasures::lnnse::lnnse_sp_3d (real(sp), dimension(:, :, :), intent(in) *x*, real(sp), dimension(:, :, :), intent(in) *y*, logical, dimension(:, :, :), intent(inout), optional *mask*)

Definition at line 1943 of file mo_errormeasures.f90.

Referenced by mo_errormeasures::lnnse_sp_3d().

Here is the caller graph for this function:

The documentation for this interface was generated from the following file:

- mo_errormeasures.f90

15.54 mo_utils::locate Interface Reference

Find closest values in a monotonic series, returns the indexes.

Public Member Functions

- integer(i4) function locate_0d_dp (x, y)
- integer(i4) function locate_0d_sp (x, y)
- integer(i4) function,
  dimension(:), allocatable locate_1d_dp (x, y)
- integer(i4) function,
  dimension(:), allocatable locate_1d_sp (x, y)

15.54.1 Detailed Description

Find closest values in a monotonic series, returns the indexes.

Given an array x(1:n), and given a value y, returns a value j such that y is between x(j) and x(j+1).
x must be monotonically increasing.
j=0 or j=N is returned to indicate that x is out of range.

Parameters

in	*real(dp/sp) :: x(:)*	Sorted array
in	*real(dp/sp) :: y[(:)]*	Value(s) of which the closest match in x(:) is wanted

Returns

integer(i4) :: index[(:)] — index(es) of x so that y is between x(index) and x(index+1)

Note

x must be monotonically increasing.

Author

Matthias Cuntz

Date

May 2014

Definition at line 264 of file mo_utils.f90.

15.54.2 Member Function/Subroutine Documentation

15.54.2.1 integer(i4) function mo_utils::locate::locate_0d_dp (real(dp), dimension(:), intent(in) *x*, real(dp), intent(in) *y*)

Definition at line 675 of file mo_utils.f90.

Referenced by mo_utils::locate_0d_dp().

Here is the caller graph for this function:

15.54.2.2 integer(i4) function mo_utils::locate::locate_0d_sp (real(sp), dimension(:), intent(in) *x*, real(sp), intent(in) *y*)

Definition at line 694 of file mo_utils.f90.

Referenced by mo_utils::locate_0d_sp().

Here is the caller graph for this function:

15.54.2.3 integer(i4) function, dimension(:), allocatable mo_utils::locate::locate_1d_dp (real(dp), dimension(:), intent(in) *x*, real(dp), dimension(:), intent(in) *y*)

Definition at line 713 of file mo_utils.f90.

Referenced by mo_utils::locate_1d_dp().

Here is the caller graph for this function:

15.54.2.4 integer(i4) function, dimension(:), allocatable mo_utils::locate::locate_1d_sp (real(sp), dimension(:), intent(in) *x*, real(sp), dimension(:), intent(in) *y*)

Definition at line 738 of file mo_utils.f90.

Referenced by mo_utils::locate_1d_sp().

Here is the caller graph for this function:

The documentation for this interface was generated from the following file:

- mo_utils.f90

15.55 mo_mad::mad Interface Reference

Public Member Functions

- logical function, dimension(size(arr)) mad_sp (arr, z, mask, deriv)
- logical function, dimension(size(arr)) mad_dp (arr, z, mask, deriv)
- real(dp) function, dimension(size(arr)) mad_val_dp (arr, z, mask, tout, mval)
- real(sp) function, dimension(size(arr)) mad_val_sp (arr, z, mask, tout, mval)

15.55.1 Detailed Description

Definition at line 110 of file mo_mad.f90.

15.55.2 Member Function/Subroutine Documentation

15.55.2.1 logical function, dimension(size(arr)) mo_mad::mad::mad_dp (real(dp), dimension(:), intent(in) *arr,* real(dp), intent(in), optional *z,* logical, dimension(:), intent(in), optional *mask,* integer(i4), intent(in), optional *deriv*)

Definition at line 189 of file mo_mad.f90.

Referenced by mo_mad::mad_dp().

Here is the caller graph for this function:

15.55.2.2 logical function, dimension(size(arr)) mo_mad::mad::mad_sp (real(sp), dimension(:), intent(in) *arr,* real(sp), intent(in), optional *z,* logical, dimension(:), intent(in), optional *mask,* integer(i4), intent(in), optional *deriv*)

Definition at line 257 of file mo_mad.f90.

Referenced by mo_mad::mad_sp().

Here is the caller graph for this function:

15.55.2.3 real(dp) function, dimension(size(arr)) mo_mad::mad::mad_val_dp (real(dp), dimension(:), intent(in) *arr,* real(dp), intent(in), optional *z,* logical, dimension(:), intent(in), optional *mask,* character(1) *tout,* real(dp), intent(in), optional *mval*)

Definition at line 326 of file mo_mad.f90.

Referenced by mo_mad::mad_val_dp().

Here is the caller graph for this function:

15.55.2.4 real(sp) function, dimension(size(arr)) mo_mad::mad::mad_val_sp (real(sp), dimension(:), intent(in) *arr,* real(sp), intent(in), optional *z,* logical, dimension(:), intent(in), optional *mask,* character(1) *tout,* real(sp), intent(in), optional *mval*)

Definition at line 389 of file mo_mad.f90.

Referenced by mo_mad::mad_val_sp().

Here is the caller graph for this function:

The documentation for this interface was generated from the following file:

- mo_mad.f90

15.56 mo_errormeasures::mae Interface Reference

Public Member Functions

- real(sp) function mae_sp_1d (x, y, mask)
- real(dp) function mae_dp_1d (x, y, mask)
- real(sp) function mae_sp_2d (x, y, mask)
- real(dp) function mae_dp_2d (x, y, mask)
- real(sp) function mae_sp_3d (x, y, mask)
- real(dp) function mae_dp_3d (x, y, mask)

15.56.1 Detailed Description

Definition at line 395 of file mo_errormeasures.f90.

15.56.2 Member Function/Subroutine Documentation

15.56.2.1 real(dp) function mo_errormeasures::mae::mae_dp_1d (real(dp), dimension(:), intent(in) *x*, real(dp), dimension(:), intent(in) *y*, logical, dimension(:), intent(in), optional *mask*)

Definition at line 2090 of file mo_errormeasures.f90.

Referenced by mo_errormeasures::mae_dp_1d().

Here is the caller graph for this function:

15.56.2.2 real(dp) function mo_errormeasures::mae::mae_dp_2d (real(dp), dimension(:, :), intent(in) *x*, real(dp), dimension(:, :), intent(in) *y*, logical, dimension(:, :), intent(in), optional *mask*)

Definition at line 2156 of file mo_errormeasures.f90.

Referenced by mo_errormeasures::mae_dp_2d().

Here is the caller graph for this function:

15.56.2.3 real(dp) function mo_errormeasures::mae::mae_dp_3d (real(dp), dimension(:, :, :), intent(in) *x*, real(dp), dimension(:, :, :), intent(in) *y*, logical, dimension(:, :, :), intent(in), optional *mask*)

Definition at line 2223 of file mo_errormeasures.f90.

Referenced by mo_errormeasures::mae_dp_3d().

Here is the caller graph for this function:

15.56.2.4 real(sp) function mo_errormeasures::mae::mae_sp_1d (real(sp), dimension(:), intent(in) *x*, real(sp), dimension(:), intent(in) *y*, logical, dimension(:), intent(in), optional *mask*)

Definition at line 2057 of file mo_errormeasures.f90.

Referenced by mo_errormeasures::mae_sp_1d().

Here is the caller graph for this function:

15.56.2.5 real(sp) function mo_errormeasures::mae::mae_sp_2d (real(sp), dimension(:, :), intent(in) *x*, real(sp), dimension(:, :), intent(in) *y*, logical, dimension(:, :), intent(in), optional *mask*)

Definition at line 2123 of file mo_errormeasures.f90.

Referenced by mo_errormeasures::mae_sp_2d().

Here is the caller graph for this function:

15.56.2.6 real(sp) function mo_errormeasures::mae::mae_sp_3d (real(sp), dimension(:, :, :), intent(in) *x*, real(sp), dimension(:, :, :), intent(in) *y*, logical, dimension(:, :, :), intent(in), optional *mask*)

Definition at line 2189 of file mo_errormeasures.f90.

Referenced by mo_errormeasures::mae_sp_3d().

Here is the caller graph for this function:

The documentation for this interface was generated from the following file:

- mo_errormeasures.f90

15.57 mo_mcmc::mcmc Interface Reference

This module is Monte Carlo Markov Chain sampling of a posterior parameter distribution (measurement errors are either known or modeled).

Public Member Functions

- subroutine mcmc_dp (eval, likelihood, para, rangePar,mcmc_paras, burnin_paras,seed_in, printflag_in, maskpara_in,restart, restart_file,tmp_file,

15.57.1 Detailed Description

This module is Monte Carlo Markov Chain sampling of a posterior parameter distribution (measurement errors are either known or modeled).

Sample posterior parameter distribution with Metropolis Algorithm.
This sampling is performed in two steps, i.e. the burn-in phase for adjusting model dependent parameters for the second step which is the proper sampling using the Metropolis Hastings Algorithm.

This sampler does not change the best parameter set, i.e. it cannot be used as an optimiser.
However, the serial and the parallel version give therefore the bitwise same results. **1. BURN IN PHASE: FIND THE OPTIMAL STEP SIZE**

Purpose:
Find optimal stepsize for each parameter such that the acceptance ratio converges to a value around 0.3.
Important variables:

Variable	Description
burnin_iter	length of markov chain performed to calculate acceptance ratio
acceptance ratio	ratio between accepted jumps and all trials (LEN)
acceptance multiplier	stepsize of a parameter is multiplied with this value when jump is

accepted (initial : 1.01) rejection multiplier | stepsize of a parameter is multiplied with this value when jump is rejected (initial : 0.99 and will never be changed) stepsize | a new parameter value is chosen based on a uniform distribution pnew_i = pold_i + Unif(-stepsize_i, stepsize_i) (initial : stepsize_i = 1.0 for all i)
Algorithm:

1. start a new markov chain of length burnin_iter with initial parameter set is the OPTIMAL one
    - select a set of parameters to change:
        - accurate –> choose one parameter,
        - comput. efficient –> choose all parameters,
        - moderate accurate & efficient –> choose half of the parameters
    - change parameter(s) based on their stepsize
    - decide whether changed parameter set is accepted or rejected:
        - $\text{oddsRatio} = \frac{\text{likelihood}(p_{new})}{\text{likelihood}(p_{old})}$
        - random number r = Uniform[0,1]
        - odds Ratio $>$ 0 –> positive accept
          odds Ratio $>$ r –> negative accept
          odds Ratio $<$ r –> reject
    - adapt stepsize of parameters changed:

  - accepted step: stepsize_i = stepsize_i $*$ accptance multiplier
  - rejected step: stepsize_i = stepsize_i $*$ rejection multiplier
- if step is accepted: for all changed parameter(s) change stepsize

2. calculate acceptance ratio of the Markov Chain
3. adjust acceptance multiplier acc_mult and store good ratios in history list
   - acceptance ratio $<$ 0.23 $-$> acc_mult = acc_mult $*$ 0.99
     delete history list
   - acceptance ratio $>$ 0.44 $-$> acc_mult = acc_mult $*$ 1.01
     delete history list
   - 0.23 $<$ acceptance ratio $<$ 0.44
     add acceptance ratio to history list
4. check if already 10 values are stored in history list and if they have converged to a value above 0.3
   (mean above 0.3 and variance less $\sqrt{1/12 * 0.05^2}$ = Variance of uniform [acc_ratio +/- 2.5%])
   - if check is positive abort and save stepsizes
     else goto (1)

2. MONTE CARLO MARKOV CHAIN: SAMPLE POSTERIOR DISTRIBUTION OF PARAMETER

Purpose:
use the previous adapted stepsizes and perform ONE monte carlo markov chain
the accepted parameter sets show the posterior distribution of parameters
Important variables:

Variable	Description
iter_mcmc	length of the markov chain ($>>$ iter_burnin)
stepsize	a new parameter value is chosen based on a uniform distribution

pnew_i = pold_i + Unif(-stepsize_i, stepsize_i) use stepsizes of the burn-in (1)

Algorithm:

1. select a set of parameters to change
   - accurate $-$> choose one parameter,
   - comput. efficient $-$> choose all parameters,

- moderate accurate & efficient –> choose half of the parameters

2. change parameter(s) based on their stepsize

3. decide whether changed parameter set is accepted or rejected:

   - $\text{oddsRatio} = \frac{\text{likelihood}(p_{new})}{\text{likelihood}(p_{old})}$

   - random number r = Uniform[0,1]

   - odds Ratio > 0 –> positive accept
     odds Ratio > r –> negative accept
     odds Ratio < r –> reject

4. if step is accepted: save parameter set

5. goto (1)

Parameters

in	*real(dp) :: likelihood(x)*	Interface Function which calculates likelihood of given parameter set x
in	*real(dp) :: para(:)*	Inital parameter set (should be GOOD approximation of best parameter set)
in	*real(dp) :: range↩ Par(size(para),2)*	Min/max range of parameters
out	*real(dp), allocatable :: mcmc_paras(:,:)*	Parameter sets sampled in proper MCMC part of algorithm
out	*real(dp), allocatable :: burnin_paras(:,:)*	Parameter sets sampled during burn-in part of algorithm
in	*integer(i8), optional :: seed_in*	User seed to initialise the random number generator (default: none –> initialized with timeseed)
in	*logical, optional :: printflag_in*	Print of output on command line (default: .False.)
in	*logical, optional :: maskpara_↩ in(size(para))*	Parameter will be sampled (.True.) or not (.False.) (default: .True.)
in	*character(len=∗), optional :: tmp_file*	filename for temporal data saving: every iter_mcmc_in iterations parameter sets are appended to this file the number of the chain will be prepended to filename output format: netcdf (default: no file writing)

in	*logical, optional :: loglike_in*	true if loglikelihood function is given instead of likelihood function (default: .false.)
in	*integer(i4), optional :: Para↩ SelectMode_in*	How many parameters will be changed at once? • half of the parameter –> 1_i4 • only one parameter –> 2_i4 • all parameter –> 3_i4 (default: 2_i4)
in	*integer(i4), optional :: iter_burnin_in*	Length of Markov chains of initial burn-in part (default: Max(250, 200∗count(maskpara)))
in	*integer(i4), optional :: iter_mcmc_in*	Length of Markov chains of proper MCMC part (default: 1000 ∗ count(maskpara))
in	*integer(i4), optional :: chains_in*	number of parallel mcmc chains (default: 5_i4)
in	*real(dp), DIMENSIO↩ N(size(para,1)), optional :: stepsize_in*	stepsize for each parameter if given burn-in is discarded (default: none –> adjusted in burn-in)

Note

Likelihood has to be defined as a function interface
The maximal number of parameters is 1000.

Definition at line 247 of file mo_mcmc.f90.

15.57.2 Member Function/Subroutine Documentation

15.57.2.1 subroutine mo_mcmc::mcmc::mcmc_dp (procedure(eval_interface), intent(in), pointer *eval,* procedure(objective_interface), intent(in), pointer *likelihood,* real(dp), dimension(:), intent(in) *para,* real(dp), dimension(:, :), intent(in) *rangePar,* real(dp), dimension(:, :), intent(out), allocatable *mcmc_paras,* real(dp), dimension(:, :), intent(out), allocatable *burnin_paras,* integer(i8), intent(in), optional *seed_in,* logical, intent(in), optional *printflag_in,* logical, dimension(size(para, 1)), intent(in), optional *maskpara_in,* logical, intent(in), optional *restart,* character(len = ∗), intent(in), optional *restart_file,* character(len = ∗), intent(in), optional *tmp_file*)

Definition at line 483 of file mo_mcmc.f90.

The documentation for this interface was generated from the following file:

- mo_mcmc.f90

15.58 mo_mcmc::mcmc_stddev Interface Reference

This module is Monte Carlo Markov Chain sampling of a posterior parameter distribution (measurement errors are neither known nor modeled).

Public Member Functions

- subroutine mcmc_stddev_dp (eval, likelihood, para, rangePar,mcmc_paras, burnin_paras,seed_in, printflag↩ _in, maskpara_in,tmp_file,

15.58.1 Detailed Description

This module is Monte Carlo Markov Chain sampling of a posterior parameter distribution (measurement errors are neither known nor modeled).

Sample posterior parameter distribution with Metropolis Algorithm.
This sampling is performed in two steps, i.e. the burn-in phase for adjusting model dependent parameters for the second step which is the proper sampling using the Metropolis Hastings Algorithm.

1. BURN IN PHASE: FIND THE OPTIMAL STEP SIZE

Purpose:
Find optimal stepsize for each parameter such that the acceptance ratio converges to a value around 0.3.
Important variables:

Variable	Description
burnin_iter	length of markov chain performed to calculate acceptance ratio
acceptance ratio	ratio between accepted jumps and all trials (LEN)
acceptance multiplier	stepsize of a parameter is multiplied with this value when jump is

accepted (initial : 1.01) rejection multiplier | stepsize of a parameter is multiplied with this value when jump is rejected (initial : 0.99 and will never be changed) stepsize | a new parameter value is chosen based on a uniform distribution pnew_i = pold_i + Unif(-stepsize_i, stepsize_i) (initial : stepsize_i = 1.0 for all i)
Algorithm:

1. start a new markov chain of length burnin_iter with initial parameter set is the OPTIMAL one
   - select a set of parameters to change:
     - accurate –> choose one parameter,
     - comput. efficient –> choose all parameters,
     - moderate accurate & efficient –> choose half of the parameters
   - change parameter(s) based on their stepsize
   - decide whether changed parameter set is accepted or rejected:
     - $\text{oddsRatio} = \frac{\text{likelihood}(p_{new})}{\text{likelihood}(p_{old})}$
     - random number r = Uniform[0,1]
     - odds Ratio $>$ 0 –> positive accept
       odds Ratio $>$ r –> negative accept
       odds Ratio $<$ r –> reject
   - adapt stepsize of parameters changed:
     - accepted step: stepsize_i = stepsize_i $*$ accptance multiplier

  - rejected step: stepsize_i = stepsize_i $*$ rejection multiplier

- if step is accepted: for all changed parameter(s) change stepsize

2. calculate acceptance ratio of the Markov Chain

3. adjust acceptance multiplier acc_mult and store good ratios in history list

   - acceptance ratio $<$ 0.23 $-$> acc_mult = acc_mult $*$ 0.99
     delete history list

   - acceptance ratio $>$ 0.44 $-$> acc_mult = acc_mult $*$ 1.01
     delete history list

   - 0.23 $<$ acceptance ratio $<$ 0.44
     add acceptance ratio to history list

4. check if already 10 values are stored in history list and if they have converged to a value above 0.3
   (mean above 0.3 and variance less $\sqrt{1/12 * 0.05^2}$ = Variance of uniform [acc_ratio +/- 2.5%])

   - if check is positive abort and save stepsizes
     else goto (1)

2. MONTE CARLO MARKOV CHAIN: SAMPLE POSTERIOR DISTRIBUTION OF PARAMETER

Purpose:
use the previous adapted stepsizes and perform ONE monte carlo markov chain
the accepted parameter sets show the posterior distribution of parameters
Important variables:

Variable	Description
iter_mcmc	length of the markov chain ($>>$ iter_burnin)
stepsize	a new parameter value is chosen based on a uniform distribution

pnew_i = pold_i + Unif(-stepsize_i, stepsize_i) use stepsizes of the burn-in (1)

Algorithm:

1. select a set of parameters to change

   - accurate $-$> choose one parameter,

   - comput. efficient $-$> choose all parameters,

   - moderate accurate & efficient $-$> choose half of the parameters

2. change parameter(s) based on their stepsize

3. decide whether changed parameter set is accepted or rejected:

   - $\text{oddsRatio} = \frac{\text{likelihood}(p_{new})}{\text{likelihood}(p_{old})}$

   - random number r = Uniform[0,1]

   - odds Ratio $>$ 0 $-$> positive accept
     odds Ratio $>$ r $-$> negative accept
     odds Ratio $<$ r $-$> reject

4. if step is accepted: save parameter set

5. goto (1)

Parameters

in	*real(dp) :: likelihood(x,sigma,stddev_new,likeli_new)*	Interface Function which calculates likelihood of given parameter set x and given standard deviation sigma and returns optionally the standard deviation stddev_new of the errors using x and likelihood likeli_new using stddev_new
in	*real(dp) :: para(:)*	Inital parameter set (should be GOOD approximation of best parameter set)
in	*real(dp) :: range↩Par(size(para),2)*	Min/max range of parameters
out	*real(dp), allocatable :: mcmc_paras(:,:)*	Parameter sets sampled in proper MCMC part of algorithm
out	*real(dp), allocatable :: burnin_paras(:,:)*	Parameter sets sampled during burn-in part of algorithm
in	*integer(i8), optional :: seed_in*	User seed to initialise the random number generator (default: none $-$> initialized with timeseed)
in	*logical, optional :: printflag_in*	Print of output on command line (default: .False.)
in	*logical, optional :: maskpara_↩in(size(para))*	Parameter will be sampled (.True.) or not (.False.) (default: .True.)
in	*character(len=∗), optional :: tmp_file*	filename for temporal data saving: every iter_mcmc_in iterations parameter sets are appended to this file the number of the chain will be prepended to filename output format: netcdf (default: no file writing)

in	*logical, optional :: loglike_in*	true if loglikelihood function is given instead of likelihood function (default: .false.)
in	*integer(i4), optional :: Para↩ SelectMode_in*	How many parameters will be changed at once? • half of the parameter –> 1_i4 • only one parameter –> 2_i4 • all parameter –> 3_i4 (default: 2_i4)
in	*integer(i4), optional :: iter_burnin_in*	Length of Markov chains of initial burn-in part (default: Max(250, 200∗count(maskpara)))
in	*integer(i4), optional :: iter_mcmc_in*	Length of Markov chains of proper MCMC part (default: 1000 ∗ count(maskpara))
in	*integer(i4), optional :: chains_in*	number of parallel mcmc chains (default: 5_i4)
in	*real(dp), DIMENSIO↩ N(size(para,1)), optional :: stepsize_in*	stepsize for each parameter if given burn-in is discarded (default: none –> adjusted in burn-in)

Definition at line 469 of file mo_mcmc.f90.

15.58.2 Member Function/Subroutine Documentation

15.58.2.1 subroutine mo_mcmc::mcmc_stddev::mcmc_stddev_dp (procedure(eval_interface), intent(in), pointer *eval,* procedure(objective_interface), intent(in), pointer *likelihood,* real(dp), dimension(:), intent(in) *para,* real(dp), dimension(:, :), intent(in) *rangePar,* real(dp), dimension(:, :), intent(out), allocatable *mcmc_paras,* real(dp), dimension(:, :), intent(out), allocatable *burnin_paras,* integer(i8), intent(in), optional *seed_in,* logical, intent(in), optional *printflag_in,* logical, dimension(size(para, 1)), intent(in), optional *maskpara_in,* character(len = ∗), intent(in), optional *tmp_file*)

Definition at line 1338 of file mo_mcmc.f90.

The documentation for this interface was generated from the following file:

- mo_mcmc.f90

15.59 mo_template::mean Interface Reference

The average.

Public Member Functions

- real(sp) function mean_sp (dat, mask)
- real(dp) function mean_dp (dat, mask)

15.59.1 Detailed Description

The average.

Calculates the average value of a vector, i.e. the first moment of a series of numbers:

$$\bar{x} = \frac{1}{N}\sum_{i=1}^{N} x_i$$

If an optinal mask is given, the mean is only over those locations that correspond to true values in the mask. x can be single or double precision. The result will have the same numerical precision. ADDITIONAL INFORMATION vec = (/ 1., 2, 3., -999., 5., 6. /) m = mean(vec, mask=(vec >= 0.)) -> see also example in test directory Sokal RR & Rohlf FJ - Biometry: the principle and practice of statistics in biological research, Freeman & Co., ISBN 0-7167-2411-1 Press WH, Teukolsky SA, Vetterling WT, & Flannery BP - Numerical Recipes in Fortran 90 - The Art of Parallel Scientific Computing, 2nd Edition, Volume 2 of Fortran Numerical Recipes, Cambridge University Press, UK, 1996

Returns

real(sp/dp) :: mean — $\bar{x}$ average of all elements in vec

Authors

Matthias Cuntz

Date

Nov 2011

Definition at line 79 of file mo_template.f90.

15.59.2 Member Function/Subroutine Documentation

15.59.2.1 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.

Referenced by mo_moment::mean_dp(), and mo_template::mean_dp().

Here is the caller graph for this function:

15.59.2.2 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.

Referenced by mo_moment::mean_sp(), and mo_template::mean_sp().

Here is the caller graph for this function:

The documentation for this interface was generated from the following file:

- mo_template.f90

15.60 mo_moment::mean Interface Reference

Public Member Functions

- real(sp) function mean_sp (dat, mask)
- real(dp) function mean_dp (dat, mask)

15.60.1 Detailed Description

Definition at line 490 of file mo_moment.f90.

15.60.2 Member Function/Subroutine Documentation

15.60.2.1 real(dp) function mo_moment::mean::mean_dp (real(dp), dimension(:), intent(in) *dat,* logical, dimension(:), intent(in), optional *mask*)

Definition at line 1340 of file mo_moment.f90.

15.60.2.2 real(sp) function mo_moment::mean::mean_sp (real(sp), dimension(:), intent(in) *dat,* logical, dimension(:), intent(in), optional *mask*)

Definition at line 1367 of file mo_moment.f90.

The documentation for this interface was generated from the following file:

- mo_moment.f90

15.61 mo_percentile::median Interface Reference

Public Member Functions

- real(sp) function median_sp (arrin, mask)
- real(dp) function median_dp (arrin, mask)

15.61.1 Detailed Description

Definition at line 88 of file mo_percentile.f90.

15.61.2 Member Function/Subroutine Documentation

15.61.2.1 real(dp) function mo_percentile::median::median_dp (real(dp), dimension(:), intent(in) *arrin,* logical, dimension(:), intent(in), optional *mask*)

Definition at line 281 of file mo_percentile.f90.

Referenced by mo_percentile::median_dp().

Here is the caller graph for this function:

15.61.2.2 real(sp) function mo_percentile::median::median_sp (real(sp), dimension(:), intent(in) *arrin,* logical, dimension(:), intent(in), optional *mask*)

Definition at line 323 of file mo_percentile.f90.

Referenced by mo_percentile::median_sp().

Here is the caller graph for this function:

The documentation for this interface was generated from the following file:

- mo_percentile.f90

15.62 mo_moment::mixed_central_moment Interface Reference

Public Member Functions

- real(sp) function mixed_central_moment_sp (x, y, r, s, mask)
- real(dp) function mixed_central_moment_dp (x, y, r, s, mask)

15.62.1 Detailed Description

Definition at line 547 of file mo_moment.f90.

15.62.2 Member Function/Subroutine Documentation

15.62.2.1 real(dp) function mo_moment::mixed_central_moment::mixed_central_moment_dp (real(dp), dimension(:), intent(in) *x*, real(dp), dimension(:), intent(in) *y*, integer(i4), intent(in) *r*, integer(i4), intent(in) *s*, logical, dimension(:), intent(in), optional *mask*)

Definition at line 1395 of file mo_moment.f90.

Referenced by mo_moment::mixed_central_moment_dp().

Here is the caller graph for this function:

15.62.2.2 real(sp) function mo_moment::mixed_central_moment::mixed_central_moment_sp (real(sp), dimension(:), intent(in) *x*, real(sp), dimension(:), intent(in) *y*, integer(i4), intent(in) *r*, integer(i4), intent(in) *s*, logical, dimension(:), intent(in), optional *mask*)

Definition at line 1445 of file mo_moment.f90.

Referenced by mo_moment::mixed_central_moment_sp().

Here is the caller graph for this function:

The documentation for this interface was generated from the following file:

- mo_moment.f90

15.63 mo_moment::mixed_central_moment_var Interface Reference

Public Member Functions

- real(sp) function mixed_central_moment_var_sp (x, y, r, s, mask)
- real(dp) function mixed_central_moment_var_dp (x, y, r, s, mask)

15.63.1 Detailed Description

Definition at line 605 of file mo_moment.f90.

15.63.2 Member Function/Subroutine Documentation

15.63.2.1 real(dp) function mo_moment::mixed_central_moment_var::mixed_central_moment_var_dp (real(dp), dimension(:), intent(in) *x*, real(dp), dimension(:), intent(in) *y*, integer(i4), intent(in) *r*, integer(i4), intent(in) *s*, logical, dimension(:), intent(in), optional *mask*)

Definition at line 1496 of file mo_moment.f90.

Referenced by mo_moment::mixed_central_moment_var_dp().

Here is the caller graph for this function:

15.63.2.2 real(sp) function mo_moment::mixed_central_moment_var::mixed_central_moment_var_sp (real(sp), dimension(:), intent(in) *x*, real(sp), dimension(:), intent(in) *y*, integer(i4), intent(in) *r*, integer(i4), intent(in) *s*, logical, dimension(:), intent(in), optional *mask*)

Definition at line 1542 of file mo_moment.f90.

Referenced by mo_moment::mixed_central_moment_var_sp().

Here is the caller graph for this function:

The documentation for this interface was generated from the following file:

- mo_moment.f90

15.64 mo_anneal Module Reference

Data Types

- interface anneal

  anneal
- interface generate_neighborhood_weight
- interface gettemperature

  GetTemperature.

Public Member Functions

- real(dp) function gettemperature_dp (paraset, cost, eval, acc_goal, prange, prange_func, samplesize, maskpara, seeds, printflag, weight, maxit, undef_funcval)
- real(dp) function pargen_anneal_dp (old, dMax, oMin, oMax, RN)
- real(dp) function pargen_dds_dp (old, perturb, oMin, oMax, RN)
- subroutine generate_neighborhood_weight_dp (truepara, cum_weight, save_state_xor, iTotalCounter, nIT↩ ERmax, neighborhood)

Private Member Functions

- real(dp) function, dimension(size(para, 1)) anneal_dp (eval, cost, para, prange, prange_func, temp, Dt, nI↩ TERmax, Len, nST, eps, acc, seeds, printflag, maskpara, weight, changeParaMode, reflectionFlag, pertub↩ FlexFlag, maxit, undef_funcval, tmp_file, funcbest, history)
- real(dp) function dchange_dp (delta, iDigit, isZero)

15.64.1 Detailed Description

Definition at line 1 of file mo_anneal.f90.

15.64.2 Member Function/Subroutine Documentation

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

Definition at line 357 of file mo_anneal.f90.

References mo_anneal::generate_neighborhood_weight::generate_neighborhood_weight_dp(), pargen_anneal_↩ dp(), and pargen_dds_dp().

Here is the call graph for this function:

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

Definition at line 1440 of file mo_anneal.f90.

Referenced by pargen_anneal_dp().

Here is the caller graph for this function:

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

Definition at line 1465 of file mo_anneal.f90.

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

Definition at line 1095 of file mo_anneal.f90.

References mo_anneal::gettemperature::gettemperature_dp(), and pargen_anneal_dp().

Here is the call graph for this function:

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

Definition at line 1373 of file mo_anneal.f90.

References dchange_dp().

Referenced by anneal_dp(), and gettemperature_dp().

Here is the call graph for this function:

Here is the caller graph for this function:

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

Definition at line 1413 of file mo_anneal.f90.

Referenced by anneal_dp().

Here is the caller graph for this function:

The documentation for this module was generated from the following file:

- mo_anneal.f90

15.65 mo_append Module Reference

Append values on existing arrays.

Data Types

- interface append

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

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

Private Member Functions

- subroutine append_i4_v_s (vec1, sca2)
- subroutine append_i4_v_v (vec1, vec2)
- subroutine append_i4_m_m (mat1, mat2, fill_value)
- subroutine append_i4_3d (mat1, mat2, fill_value)
- subroutine append_i8_v_s (vec1, sca2)
- subroutine append_i8_v_v (vec1, vec2)
- subroutine append_i8_m_m (mat1, mat2, fill_value)
- subroutine append_i8_3d (mat1, mat2, fill_value)
- subroutine append_sp_v_s (vec1, sca2)
- subroutine append_sp_v_v (vec1, vec2)
- subroutine append_sp_m_m (mat1, mat2, fill_value)
- subroutine append_sp_3d (mat1, mat2, fill_value)
- subroutine append_dp_v_s (vec1, sca2)
- subroutine append_dp_v_v (vec1, vec2)
- subroutine append_dp_m_m (mat1, mat2, fill_value)
- subroutine append_dp_3d (mat1, mat2, fill_value)
- subroutine append_char_v_s (vec1, sca2)
- subroutine append_char_v_v (vec1, vec2)
- subroutine append_char_m_m (mat1, mat2, fill_value)
- subroutine append_char_3d (mat1, mat2, fill_value)
- subroutine append_lgt_v_s (vec1, sca2)
- subroutine append_lgt_v_v (vec1, vec2)
- subroutine append_lgt_m_m (mat1, mat2, fill_value)

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

15.65.1 Detailed Description

Append values on existing arrays.

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

Author

Juliane Mai

Date

Aug 2012

Definition at line 11 of file mo_append.f90.

15.65.2 Member Function/Subroutine Documentation

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

Definition at line 1022 of file mo_append.f90.

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

Definition at line 963 of file mo_append.f90.

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

Definition at line 901 of file mo_append.f90.

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

Definition at line 933 of file mo_append.f90.

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

Definition at line 852 of file mo_append.f90.

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

Definition at line 793 of file mo_append.f90.

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

Definition at line 729 of file mo_append.f90.

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

Definition at line 761 of file mo_append.f90.

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

Definition at line 336 of file mo_append.f90.

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

Definition at line 277 of file mo_append.f90.

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

Definition at line 213 of file mo_append.f90.

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

Definition at line 245 of file mo_append.f90.

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

Definition at line 508 of file mo_append.f90.

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

Definition at line 449 of file mo_append.f90.

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

Definition at line 385 of file mo_append.f90.

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

Definition at line 417 of file mo_append.f90.

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

Definition at line 1194 of file mo_append.f90.

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

Definition at line 1135 of file mo_append.f90.

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

Definition at line 1071 of file mo_append.f90.

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

Definition at line 1103 of file mo_append.f90.

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

Definition at line 680 of file mo_append.f90.

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

Definition at line 621 of file mo_append.f90.

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

Definition at line 557 of file mo_append.f90.

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

Definition at line 589 of file mo_append.f90.

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

Definition at line 1937 of file mo_append.f90.

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

Definition at line 1845 of file mo_append.f90.

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

Definition at line 1879 of file mo_append.f90.

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

Definition at line 1787 of file mo_append.f90.

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

Definition at line 1695 of file mo_append.f90.

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

Definition at line 1729 of file mo_append.f90.

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

Definition at line 1337 of file mo_append.f90.

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

Definition at line 1245 of file mo_append.f90.

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

Definition at line 1279 of file mo_append.f90.

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

Definition at line 1487 of file mo_append.f90.

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

Definition at line 1395 of file mo_append.f90.

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

Definition at line 1429 of file mo_append.f90.

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

Definition at line 2069 of file mo_append.f90.

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

Definition at line 1995 of file mo_append.f90.

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

Definition at line 2029 of file mo_append.f90.

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

Definition at line 1637 of file mo_append.f90.

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

Definition at line 1545 of file mo_append.f90.

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

Definition at line 1579 of file mo_append.f90.

The documentation for this module was generated from the following file:

- mo_append.f90

15.66 mo_canopy_interc Module Reference

Canopy interception.

Public Member Functions

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

15.66.1 Detailed Description

Canopy interception.

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

Authors

Vladyslav Prykhodko

Date

Dec 2012

Definition at line 14 of file mo_canopy_interc.f90.

15.66.2 Member Function/Subroutine Documentation

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

Canopy interception.

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

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

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

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

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

Parameters

in	*REAL(dp) :: pet*	Potential evapotranspiration [mm s-1]
in	*REAL(dp) :: interc_max*	Maximum interception [mm]
in	*REAL(dp) :: precip*	Daily mean precipitation [mm]

in,out	*REAL(dp) :: interc*	Interception [mm]
out	*REAL(dp) :: throughfall*	Throughfall [mm s-1]
out	*REAL(dp) :: evap_canopy*	Real evaporation intensity from canopy[mm s-1]

Authors

Vladyslav Prykhodko

Date

Dec 2012

Definition at line 74 of file mo_canopy_interc.f90.

Referenced by mo_mhm::mhm().

Here is the caller graph for this function:

The documentation for this module was generated from the following file:

- mo_canopy_interc.f90

15.67 mo_common_constants Module Reference

Provides constants commonly used by mHM, mRM and MPR.

Collaboration diagram for mo_common_constants:

Public Attributes

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

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

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

15.67.1 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

Definition at line 14 of file mo_common_constants.f90.

15.67.2 Member Data Documentation

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

Definition at line 41 of file mo_common_constants.f90.

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

Definition at line 45 of file mo_common_constants.f90.

15.67.2.3 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.

15.67.2.4 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.

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

Definition at line 46 of file mo_common_constants.f90.

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

Definition at line 38 of file mo_common_constants.f90.

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

Definition at line 37 of file mo_common_constants.f90.

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

Definition at line 36 of file mo_common_constants.f90.

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

Definition at line 30 of file mo_common_constants.f90.

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

Definition at line 29 of file mo_common_constants.f90.

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

Definition at line 33 of file mo_common_constants.f90.

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

Definition at line 44 of file mo_common_constants.f90.

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

Definition at line 42 of file mo_common_constants.f90.

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

Definition at line 43 of file mo_common_constants.f90.

The documentation for this module was generated from the following file:

- mo_common_constants.f90

15.68 mo_common_file Module Reference

Provides file names and units for mRM.

Collaboration diagram for mo_common_file:

Public Attributes

- 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.

15.68.1 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

Definition at line 15 of file mo_common_file.f90.

15.68.2 Member Data Documentation

15.68.2.1 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.

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

DEM input data file.

Definition at line 19 of file mo_common_file.f90.

15.68.2.3 integer, parameter mo_common_file::uconfig = 68

Unit for file defining mHM's outputs.

Definition at line 28 of file mo_common_file.f90.

15.68.2.4 integer, parameter mo_common_file::udem = 53

Unit for DEM input data file.

Definition at line 21 of file mo_common_file.f90.

15.68.2.5 integer, parameter mo_common_file::ulcoverclass = 61

Unit for LCover input data file.

Definition at line 23 of file mo_common_file.f90.

The documentation for this module was generated from the following file:

- mo_common_file.f90

15.69 mo_common_functions Module Reference

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

Public Member Functions

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

15.69.1 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

Definition at line 14 of file mo_common_functions.f90.

15.69.2 Member Function/Subroutine Documentation

15.69.2.1 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_mrm_read_config::read_mrm_routing_params().

Here is the caller graph for this function:

The documentation for this module was generated from the following file:

- mo_common_functions.f90

15.70 mo_common_read_config Module Reference

Reading of main model configurations.

Public Member Functions

- subroutine, public common_read_config (file_namelist, unamelist)

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

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

15.70.1 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

Definition at line 13 of file mo_common_read_config.f90.

15.70.2 Member Function/Subroutine Documentation

15.70.2.1 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_nml::close_nml(), mo_message::message(), mo_nml::open_nml(), and mo_nml::position_nml().

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

Here is the call graph for this function:

Here is the caller graph for this function:

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

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

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

Parameters

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

in,out	*character(256), dimension(:) :: LCfilename*	

Authors

Matthias Zink

Date

Dec 2012

Definition at line 234 of file mo_common_read_config.f90.

References mo_message::message().

Here is the call graph for this function:

The documentation for this module was generated from the following file:

- mo_common_read_config.f90

15.71 mo_common_read_data Module Reference

TODO: add description.

Public Member Functions

- subroutine, public read_dem
  TODO: add description.
- subroutine, public read_lcover
  TODO: add description.

15.71.1 Detailed Description

TODO: add description.

TODO: add description

Authors

Robert Schweppe

Date

Jun 2018

Definition at line 13 of file mo_common_read_data.f90.

15.71.2 Member Function/Subroutine Documentation

15.71.2.1 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_message::message(), mo_read_spatial_data::read_header_ascii(), and mo_grid::set_basin_↩indices().

Referenced by mo_mrm_read_data::mrm_read_l0_data().

Here is the call graph for this function:

Here is the caller graph for this function:

15.71.2.2 subroutine, public mo_common_read_data::read_lcover ( )

TODO: add description.

TODO: add description

Authors

Robert Schweppe

Date

Jun 2018

Definition at line 142 of file mo_common_read_data.f90.

References mo_message::message().

Referenced by mo_mrm_read_data::mrm_read_l0_data().

Here is the call graph for this function:

Here is the caller graph for this function:

The documentation for this module was generated from the following file:

- mo_common_read_data.f90

15.72 mo_common_restart Module Reference

TODO: add description.

Public Member Functions

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

  write restart files for each basin

- subroutine, public read_grid_info (iBasin, InPath, level_name, fname_part, new_grid)

  reads configuration apart from Level 11 configuration from a restart directory

15.72.1 Detailed Description

TODO: add description.

TODO: add description

Authors

Robert Schweppe

Date

Jun 2018

Definition at line 13 of file mo_common_restart.f90.

15.72.2 Member Function/Subroutine Documentation

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

reads configuration apart from Level 11 configuration from a restart directory

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

Parameters

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

Authors

Stephan Thober

Date

Apr 2013

Definition at line 153 of file mo_common_restart.f90.

References mo_message::message().

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

Here is the call graph for this function:

Here is the caller graph for this function:

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

write restart files for each basin

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

Parameters

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

Authors

Stephan Thober

Date

Jun 2014

Definition at line 61 of file mo_common_restart.f90.

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

Here is the caller graph for this function:

The documentation for this module was generated from the following file:

- mo_common_restart.f90

15.73 mo_common_variables Module Reference

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

Collaboration diagram for mo_common_variables:

Data Types

- type grid
- type gridremapper
- type period

Public Attributes

- character(1024), public project_details
- character(1024), public setup_description
- character(1024), public simulation_type
- character(256), public conventions
- character(1024), public contact
- character(1024), public mhm_details
- character(1024), public history
- integer(i4), public iflag_cordinate_sys
- real(dp), dimension(:), allocatable, public resolutionhydrology
- integer(i4), dimension(:), allocatable, public l0_basin
- integer(i4), dimension(:), allocatable, public l11_basin
- logical, public write_restart
- character(256), dimension(:), allocatable, public dirrestartout
- character(256), public dirconfigout
- character(256), public dircommonfiles
- character(256), dimension(:), allocatable, public dirmorpho
- character(256), dimension(:), allocatable, public dirlcover
- character(256), dimension(:), allocatable, public dirout
- character(256), dimension(:), allocatable, public filelatlon
- type(grid), dimension(:), allocatable, target, public level0
- type(grid), dimension(:), allocatable, target, public level1
- type(gridremapper), dimension(:), allocatable, public l0_l1_remap
- type(gridremapper), dimension(:), allocatable, public l0_l11_remap
- type(gridremapper), dimension(:), allocatable, public l1_l11_remap
- real(dp), dimension(:), allocatable, public l0_elev
- integer(i4), dimension(:,:), allocatable, public l0_lcover
- integer(i4), public nbasins
- integer(i4), public nuniquel0basins
- integer(i4), public nlcoverscene
- character(256), dimension(:), allocatable, public lcfilename
- integer(i4), dimension(:), allocatable, public lc_year_start

- integer(i4), dimension(:), allocatable, public lc_year_end
- integer(i4), parameter, public nprocesses = 10
- integer(i4), dimension(nprocesses, 3), public processmatrix
- real(dp), dimension(:,:), allocatable, target, public global_parameters
- character(256), dimension(:), allocatable, public global_parameters_name
- logical alma_convention

15.73.1 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

Definition at line 18 of file mo_common_variables.f90.

15.73.2 Member Data Documentation

15.73.2.1 logical mo_common_variables::alma_convention

Definition at line 177 of file mo_common_variables.f90.

15.73.2.2 character(1024), public mo_common_variables::contact

Definition at line 30 of file mo_common_variables.f90.

15.73.2.3 character(256), public mo_common_variables::conventions

Definition at line 29 of file mo_common_variables.f90.

15.73.2.4 character(256), public mo_common_variables::dircommonfiles

Definition at line 49 of file mo_common_variables.f90.

15.73.2.5 character(256), public mo_common_variables::dirconfigout

Definition at line 48 of file mo_common_variables.f90.

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

Definition at line 51 of file mo_common_variables.f90.

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

Definition at line 50 of file mo_common_variables.f90.

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

Definition at line 52 of file mo_common_variables.f90.

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

Definition at line 47 of file mo_common_variables.f90.

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

Definition at line 53 of file mo_common_variables.f90.

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

Definition at line 165 of file mo_common_variables.f90.

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

Definition at line 169 of file mo_common_variables.f90.

15.73.2.13 character(1024), public mo_common_variables::history

Definition at line 32 of file mo_common_variables.f90.

15.73.2.14 integer(i4), public mo_common_variables::iflag_cordinate_sys

Definition at line 37 of file mo_common_variables.f90.

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

Definition at line 39 of file mo_common_variables.f90.

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

Definition at line 121 of file mo_common_variables.f90.

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

Definition at line 113 of file mo_common_variables.f90.

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

Definition at line 112 of file mo_common_variables.f90.

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

Definition at line 123 of file mo_common_variables.f90.

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

Definition at line 40 of file mo_common_variables.f90.

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

Definition at line 114 of file mo_common_variables.f90.

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

Definition at line 140 of file mo_common_variables.f90.

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

Definition at line 139 of file mo_common_variables.f90.

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

Definition at line 138 of file mo_common_variables.f90.

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

Definition at line 95 of file mo_common_variables.f90.

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

Definition at line 96 of file mo_common_variables.f90.

15.73.2.27 character(1024), public mo_common_variables::mhm_details

Definition at line 31 of file mo_common_variables.f90.

15.73.2.28 integer(i4), public mo_common_variables::nbasins

Definition at line 130 of file mo_common_variables.f90.

15.73.2.29 integer(i4), public mo_common_variables::nlcoverscene

Definition at line 137 of file mo_common_variables.f90.

15.73.2.30 integer(i4), parameter, public mo_common_variables::nprocesses = 10

Definition at line 145 of file mo_common_variables.f90.

15.73.2.31 integer(i4), public mo_common_variables::nuniquel0basins

Definition at line 131 of file mo_common_variables.f90.

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

Definition at line 156 of file mo_common_variables.f90.

15.73.2.33 character(1024), public mo_common_variables::project_details

Definition at line 26 of file mo_common_variables.f90.

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

Definition at line 38 of file mo_common_variables.f90.

15.73.2.35 character(1024), public mo_common_variables::setup_description

Definition at line 27 of file mo_common_variables.f90.

15.73.2.36 character(1024), public mo_common_variables::simulation_type

Definition at line 28 of file mo_common_variables.f90.

15.73.2.37 logical, public mo_common_variables::write_restart

Definition at line 41 of file mo_common_variables.f90.

The documentation for this module was generated from the following file:

- mo_common_variables.f90

15.74 mo_constants Module Reference

Provides computational, mathematical, physical, and file constants.

Collaboration diagram for mo_constants:

Public Attributes

- real(dp), parameter pi_dp = 3.14159265358979323846264338327950288419 7_dp

  Pi in double precision.
- real(sp), parameter pi_sp = 3.14159265358979323846264338327950288419 7_sp

  Pi in single precision.
- real(dp), parameter pio2_dp = 1.57079632679489661923132169163975144209858_dp

  Pi/2 in double precision.
- real(sp), parameter pio2_sp = 1.57079632679489661923132169163975144209858_sp

  Pi/2 in single precision.
- real(dp), parameter twopi_dp = 6.28318530717958647692528676655900576839 4_dp

  2∗Pi in double precision
- real(sp), parameter twopi_sp = 6.28318530717958647692528676655900576839 4_sp

  2∗Pi in single precision
- real(dp), parameter sqrt2_dp = 1.41421356237309504880168872420969807856967_dp

  Square root of 2 in double precision.
- real(sp), parameter sqrt2_sp = 1.41421356237309504880168872420969807856967_sp

  Square root of 2 in single precision.
- real(dp), parameter twothird_dp = 0.66666666666666666666666666666666666667_dp

  2/3 in double precision
- real(sp), parameter twothird_sp = 0.66666666666666666666666666666666666667_sp

  2/3 in single precision
- real(dp), parameter deg2rad_dp = PI_dp / 180._dp

  degree to radian conversion (pi/180) in double precision
- real(sp), parameter deg2rad_sp = PI_sp / 180._sp

  degree to radian conversion (pi/180) in double precision
- real(dp), parameter rad2deg_dp = 180._dp / PI_dp

  radian to conversion (180/pi) in double precision
- real(sp), parameter rad2deg_sp = 180._sp / PI_sp

  radian to degree conversion (180/pi) in single precision

- real(sp), parameter, public secday_sp = 86400.0_sp

  Time conversion Seconds per day [s] in single precision.
- real(dp), parameter, public secday_dp = 86400.0_dp
- real(dp), parameter, public dayhours = 24.0_dp
- real(dp), parameter, public yearmonths = 12.0_dp
- real(dp), parameter, public yeardays = 365.0_dp
- real(dp), parameter, public daysecs = 86400.0_dp
- real(dp), parameter psychro_dp = 0.0646_dp

  Psychrometric constant [kPa K^-1] in double precision.
- real(sp), parameter psychro_sp = 0.0646_sp

  Psychrometric constant [kPa K^-1] in sibgle precision.
- real(dp), parameter gravity_dp = 9.81_dp

  Gravity accelaration [m^2 s^-1] in double precision.
- real(sp), parameter gravity_sp = 9.81_sp

  Gravity accelaration [m^2 s^-1] in single precision.
- real(dp), parameter solarconst_dp = 1367._dp

  Solar constant in [J m^-2 s^-1] in double precision.
- real(sp), parameter solarconst_sp = 1367._sp

  Solar constant in [J m^-2 s^-1] in single precision.
- real(dp), parameter specheatet_dp = 2.45e06_dp

  Specific heat for vaporization of water in [J m-2 mm-1] in double precision.
- real(sp), parameter specheatet_sp = 2.45e06_sp

  Specific heat for vaporization of water in [J m-2 mm-1] in single precision.
- real(dp), parameter t0_dp = 273.15_dp

  Standard temperature [K] in double precision.
- real(sp), parameter t0_sp = 273.15_sp

  Standard temperature [K] in single precision.
- real(dp), parameter sigma_dp = 5.67e-08_dp

  Stefan-Boltzmann constant [W m^-2 K^-4] in double precision.
- real(sp), parameter sigma_sp = 5.67e-08_sp

  Stefan-Boltzmann constant [W m^-2 K^-4] in single precision.
- real(sp), parameter radiusearth_sp = 6371228._sp
- real(dp), parameter radiusearth_dp = 6371228._dp
- real(dp), parameter p0_dp = 101325._dp

  standard atmospehere Standard pressure [Pa] in double precision
- real(sp), parameter p0_sp = 101325._sp

  Standard pressure [Pa] in single precision.
- real(dp), parameter rho0_dp = 1.225_dp

  standard density [kg m^-3] in double precision
- real(sp), parameter rho0_sp = 1.225_sp

  standard density [kg m^-3] in single precision
- real(dp), parameter cp0_dp = 1005.0_dp

  specific heat capacity of air [J kg^-1 K^-1] in double precision
- real(sp), parameter cp0_sp = 1005.0_sp

  specific heat capacity of air [J kg^-1 K^-1] in single precision
- real(dp), parameter pi_d = 3.14159265358979323846264338327950288419 7_dp

  Pi in double precision.
- real(sp), parameter pi = 3.14159265358979323846264338327950288419 7_sp

  Pi in single precision.
- real(dp), parameter pio2_d = 1.57079632679489661923132169163975144209858_dp

  Pi/2 in double precision.

- real(sp), parameter pio2 = 1.5707963267948966192313216916397514420985 8_sp

  Pi/2 in single precision.
- real(dp), parameter twopi_d = 6.2831853071795864769252867665590057683 94_dp

  2∗Pi in double precision
- real(sp), parameter twopi = 6.2831853071795864769252867665590057683 94_sp

  2∗Pi in single precision
- real(dp), parameter sqrt2_d = 1.4142135623730950488016887242096980785 6967_dp

  Square root of 2 in double precision.
- real(sp), parameter sqrt2 = 1.4142135623730950488016887242096980785 6967_sp

  Square root of 2 in single precision.
- real(dp), parameter euler_d = 0.5772156649015328606065120900824024310 422_dp

  Euler"s constant in double precision.
- real(sp), parameter euler = 0.5772156649015328606065120900824024310 422_sp

  Euler"s constant in single precision.
- integer, parameter nin = input_unit

  Standard input file unit.
- integer, parameter nout = output_unit

  Standard output file unit.
- integer, parameter nerr = error_unit

  Standard error file unit.
- integer, parameter nnml = 100

  Standard file unit for namelist.

15.74.1 Detailed Description

Provides computational, mathematical, physical, and file constants.

Provides computational constants like epsilon, mathematical constants such as Pi, physical constants such as the Stefan-Boltzmann constant, and file units for some standard streams such as standard in.

Author

Matthias Cuntz

Date

Nov 2011

Definition at line 12 of file mo_constants.f90.

15.74.2 Member Data Documentation

15.74.2.1 real(dp), parameter mo_constants::cp0_dp = 1005.0_dp

specific heat capacity of air [J kg^-1 K^-1] in double precision

Definition at line 123 of file mo_constants.f90.

15.74.2.2 real(sp), parameter mo_constants::cp0_sp = 1005.0_sp

specific heat capacity of air [J kg^-1 K^-1] in single precision

Definition at line 125 of file mo_constants.f90.

15.74.2.3 real(dp), parameter, public mo_constants::dayhours = 24.0_dp

Definition at line 78 of file mo_constants.f90.

15.74.2.4 real(dp), parameter, public mo_constants::daysecs = 86400.0_dp

Definition at line 81 of file mo_constants.f90.

15.74.2.5 real(dp), parameter mo_constants::deg2rad_dp = PI_dp / 180._dp

degree to radian conversion (pi/180) in double precision

Definition at line 66 of file mo_constants.f90.

15.74.2.6 real(sp), parameter mo_constants::deg2rad_sp = PI_sp / 180._sp

degree to radian conversion (pi/180) in double precision

Definition at line 68 of file mo_constants.f90.

15.74.2.7 real(sp), parameter mo_constants::euler = 0.57721566490153286060651209008240243104 22_sp

Euler"s constant in single precision.

Definition at line 147 of file mo_constants.f90.

15.74.2.8 real(dp), parameter mo_constants::euler_d = 0.57721566490153286060651209008240243104 22_dp

Euler"s constant in double precision.

Definition at line 145 of file mo_constants.f90.

15.74.2.9 real(dp), parameter mo_constants::gravity_dp = 9.81_dp

Gravity accelaration [m^2 s^-1] in double precision.

Definition at line 89 of file mo_constants.f90.

15.74.2.10 real(sp), parameter mo_constants::gravity_sp = 9.81_sp

Gravity accelaration [m^2 s^-1] in single precision.

Definition at line 91 of file mo_constants.f90.

15.74.2.11 integer, parameter mo_constants::nerr = error_unit

Standard error file unit.

Definition at line 158 of file mo_constants.f90.

15.74.2.12 integer, parameter mo_constants::nin = input_unit

Standard input file unit.

Definition at line 152 of file mo_constants.f90.

15.74.2.13 integer, parameter mo_constants::nnml = 100

Standard file unit for namelist.

Definition at line 160 of file mo_constants.f90.

15.74.2.14 integer, parameter mo_constants::nout = output_unit

Standard output file unit.

Definition at line 155 of file mo_constants.f90.

15.74.2.15 real(dp), parameter mo_constants::p0_dp = 101325._dp

standard atmospehere Standard pressure [Pa] in double precision

Definition at line 115 of file mo_constants.f90.

15.74.2.16 real(sp), parameter mo_constants::p0_sp = 101325._sp

Standard pressure [Pa] in single precision.

Definition at line 117 of file mo_constants.f90.

15.74.2.17 real(sp), parameter mo_constants::pi = 3.14159265358979323846264338327950288419 7_sp

Pi in single precision.

Definition at line 131 of file mo_constants.f90.

15.74.2.18 real(dp), parameter mo_constants::pi_d = 3.14159265358979323846264338327950288419 7_dp

Pi in double precision.

Definition at line 129 of file mo_constants.f90.

15.74.2.19 real(dp), parameter mo_constants::pi_dp = 3.14159265358979323846264338327950288419 7_dp

Pi in double precision.

Definition at line 46 of file mo_constants.f90.

15.74.2.20 real(sp), parameter mo_constants::pi_sp = 3.14159265358979323846264338327950288419 7_sp

Pi in single precision.

Definition at line 48 of file mo_constants.f90.

15.74.2.21 real(sp), parameter mo_constants::pio2 = 1.57079632679489661923132169163975144209858_sp

Pi/2 in single precision.

Definition at line 135 of file mo_constants.f90.

15.74.2.22 real(dp), parameter mo_constants::pio2_d = 1.5707963267948966192313216916397514420985 8_dp

Pi/2 in double precision.

Definition at line 133 of file mo_constants.f90.

15.74.2.23 real(dp), parameter mo_constants::pio2_dp = 1.5707963267948966192313216916397514420985 8_dp

Pi/2 in double precision.

Definition at line 50 of file mo_constants.f90.

15.74.2.24 real(sp), parameter mo_constants::pio2_sp = 1.5707963267948966192313216916397514420985 8_sp

Pi/2 in single precision.

Definition at line 52 of file mo_constants.f90.

15.74.2.25 real(dp), parameter mo_constants::psychro_dp = 0.0646_dp

Psychrometric constant [kPa K^-1] in double precision.

Definition at line 85 of file mo_constants.f90.

15.74.2.26 real(sp), parameter mo_constants::psychro_sp = 0.0646_sp

Psychrometric constant [kPa K^-1] in sibgle precision.

Definition at line 87 of file mo_constants.f90.

15.74.2.27 real(dp), parameter mo_constants::rad2deg_dp = 180._dp / PI_dp

radian to conversion (180/pi) in double precision

Definition at line 70 of file mo_constants.f90.

15.74.2.28 real(sp), parameter mo_constants::rad2deg_sp = 180._sp / PI_sp

radian to degree conversion (180/pi) in single precision

Definition at line 72 of file mo_constants.f90.

15.74.2.29 real(dp), parameter mo_constants::radiusearth_dp = 6371228._dp

Definition at line 111 of file mo_constants.f90.

15.74.2.30 real(sp), parameter mo_constants::radiusearth_sp = 6371228._sp

Definition at line 109 of file mo_constants.f90.

15.74.2.31 real(dp), parameter mo_constants::rho0_dp = 1.225_dp

standard density [kg m^-3] in double precision

Definition at line 119 of file mo_constants.f90.

15.74.2.32 real(sp), parameter mo_constants::rho0_sp = 1.225_sp

standard density [kg m^-3] in single precision

Definition at line 121 of file mo_constants.f90.

15.74.2.33 real(dp), parameter, public mo_constants::secday_dp = 86400.0_dp

Definition at line 77 of file mo_constants.f90.

15.74.2.34 real(sp), parameter, public mo_constants::secday_sp = 86400.0_sp

Time conversion Seconds per day [s] in single precision.

Definition at line 76 of file mo_constants.f90.

15.74.2.35 real(dp), parameter mo_constants::sigma_dp = 5.67e-08_dp

Stefan-Boltzmann constant [W m^-2 K^-4] in double precision.

Definition at line 105 of file mo_constants.f90.

15.74.2.36 real(sp), parameter mo_constants::sigma_sp = 5.67e-08_sp

Stefan-Boltzmann constant [W m^-2 K^-4] in single precision.

Definition at line 107 of file mo_constants.f90.

15.74.2.37 real(dp), parameter mo_constants::solarconst_dp = 1367._dp

Solar constant in [J m^-2 s^-1] in double precision.

Definition at line 93 of file mo_constants.f90.

15.74.2.38 real(sp), parameter mo_constants::solarconst_sp = 1367._sp

Solar constant in [J m^-2 s^-1] in single precision.

Definition at line 95 of file mo_constants.f90.

15.74.2.39 real(dp), parameter mo_constants::specheatet_dp = 2.45e06_dp

Specific heat for vaporization of water in [J m-2 mm-1] in double precision.

Definition at line 97 of file mo_constants.f90.

15.74.2.40 real(sp), parameter mo_constants::specheatet_sp = 2.45e06_sp

Specific heat for vaporization of water in [J m-2 mm-1] in single precision.

Definition at line 99 of file mo_constants.f90.

15.74.2.41 real(sp), parameter mo_constants::sqrt2 = 1.41421356237309504880168872420969807856967_sp

Square root of 2 in single precision.

Definition at line 143 of file mo_constants.f90.

15.74.2.42 real(dp), parameter mo_constants::sqrt2_d = 1.41421356237309504880168872420969807856967_dp

Square root of 2 in double precision.

Definition at line 141 of file mo_constants.f90.

15.74.2.43 real(dp), parameter mo_constants::sqrt2_dp = 1.41421356237309504880168872420969807856967_dp

Square root of 2 in double precision.

Definition at line 58 of file mo_constants.f90.

15.74.2.44 real(sp), parameter mo_constants::sqrt2_sp = 1.41421356237309504880168872420969807856967_sp

Square root of 2 in single precision.

Definition at line 60 of file mo_constants.f90.

15.74.2.45 real(dp), parameter mo_constants::t0_dp = 273.15_dp

Standard temperature [K] in double precision.

Definition at line 101 of file mo_constants.f90.

15.74.2.46 real(sp), parameter mo_constants::t0_sp = 273.15_sp

Standard temperature [K] in single precision.

Definition at line 103 of file mo_constants.f90.

15.74.2.47 real(sp), parameter mo_constants::twopi = 6.28318530717958647692528676655900576839 4_sp

2∗Pi in single precision

Definition at line 139 of file mo_constants.f90.

15.74.2.48 real(dp), parameter mo_constants::twopi_d = 6.28318530717958647692528676655900576839 4_dp

2∗Pi in double precision

Definition at line 137 of file mo_constants.f90.

15.74.2.49 real(dp), parameter mo_constants::twopi_dp = 6.28318530717958647692528676655900576839 4_dp

2∗Pi in double precision

Definition at line 54 of file mo_constants.f90.

15.74.2.50 real(sp), parameter mo_constants::twopi_sp = 6.2831853071795864769252867665590057683943_sp

2∗Pi in single precision

Definition at line 56 of file mo_constants.f90.

15.74.2.51 real(dp), parameter mo_constants::twothird_dp = 0.6666666666666666666666666666666666667_dp

2/3 in double precision

Definition at line 62 of file mo_constants.f90.

15.74.2.52 real(sp), parameter mo_constants::twothird_sp = 0.6666666666666666666666666666666666667_sp

2/3 in single precision

Definition at line 64 of file mo_constants.f90.

15.74.2.53 real(dp), parameter, public mo_constants::yeardays = 365.0_dp

Definition at line 80 of file mo_constants.f90.

15.74.2.54 real(dp), parameter, public mo_constants::yearmonths = 12.0_dp

Definition at line 79 of file mo_constants.f90.

The documentation for this module was generated from the following file:

- mo_constants.f90

15.75 mo_corr Module Reference

Collaboration diagram for mo_corr:

Data Types

- interface arth

- interface autocoeffk
- interface autocorr
- interface corr
- interface crosscoeffk
- interface crosscorr
- interface four1
- interface fourrow
- interface realft
- interface swap

Private Member Functions

- real(sp) function, dimension(n) arth_sp (first, increment, n)
- real(dp) function, dimension(n) arth_dp (first, increment, n)
- integer(i4) function, dimension(n) arth_i4 (first, increment, n)
- real(dp) function autocoeffk_dp (x, k, mask)
- real(sp) function autocoeffk_sp (x, k, mask)
- real(dp) function, dimension(size(k)) autocoeffk_1d_dp (x, k, mask)
- real(sp) function, dimension(size(k)) autocoeffk_1d_sp (x, k, mask)
- real(dp) function autocorr_dp (x, k, mask)
- real(sp) function autocorr_sp (x, k, mask)
- real(dp) function, dimension(size(k)) autocorr_1d_dp (x, k, mask)
- real(sp) function, dimension(size(k)) autocorr_1d_sp (x, k, mask)
- real(dp) function, dimension(size(data1)) corr_dp (data1, data2, nadjust, nhigh, nwin)
- real(sp) function, dimension(size(data1)) corr_sp (data1, data2, nadjust, nhigh, nwin)
- real(dp) function crosscoeffk_dp (x, y, k, mask)
- real(sp) function crosscoeffk_sp (x, y, k, mask)
- real(dp) function crosscorr_dp (x, y, k, mask)
- real(sp) function crosscorr_sp (x, y, k, mask)
- subroutine four1_sp (data, isign)
- subroutine four1_dp (data, isign)
- subroutine fourrow_sp (data, isign)
- subroutine fourrow_dp (data, isign)
- subroutine realft_dp (data, isign, zdata)
- subroutine realft_sp (data, isign, zdata)
- subroutine swap_1d_spc (a, b)
- subroutine swap_1d_dpc (a, b)
- complex(dpc) function,
  dimension(nn) zroots_unity_dp (n, nn)
- complex(spc) function,
  dimension(nn) zroots_unity_sp (n, nn)

Private Attributes

- integer(i4), parameter npar_arth = 16
- integer(i4), parameter npar2_arth = 8

15.75.1 Detailed Description

Definition at line 1 of file mo_corr.f90.

15.75.2 Member Function/Subroutine Documentation

15.75.2.1 real(dp) function, dimension(n) mo_corr::arth_dp (real(dp), intent(in) *first,* real(dp), intent(in) *increment,* integer(i4), intent(in) *n*) [private]

Definition at line 453 of file mo_corr.f90.

References mo_corr::arth::arth_dp().

Here is the call graph for this function:

15.75.2.2 integer(i4) function, dimension(n) mo_corr::arth_i4 (integer(i4), intent(in) *first,* integer(i4), intent(in) *increment,* integer(i4), intent(in) *n*) [private]

Definition at line 487 of file mo_corr.f90.

References mo_corr::arth::arth_i4().

Here is the call graph for this function:

15.75.2.3 real(sp) function, dimension(n) mo_corr::arth_sp (real(sp), intent(in) *first,* real(sp), intent(in) *increment,* integer(i4), intent(in) *n*) [private]

Definition at line 419 of file mo_corr.f90.

References mo_corr::arth::arth_sp().

Here is the call graph for this function:

15.75.2.4 real(dp) function, dimension(size(k)) mo_corr::autocoeffk_1d_dp (real(dp), dimension(:), intent(in) *x*, integer(i4), dimension(:), intent(in) *k*, logical, dimension(:), intent(in), optional *mask*) [private]

Definition at line 557 of file mo_corr.f90.

15.75.2.5 real(sp) function, dimension(size(k)) mo_corr::autocoeffk_1d_sp (real(sp), dimension(:), intent(in) *x*, integer(i4), dimension(:), intent(in) *k*, logical, dimension(:), intent(in), optional *mask*) [private]

Definition at line 581 of file mo_corr.f90.

15.75.2.6 real(dp) function mo_corr::autocoeffk_dp (real(dp), dimension(:), intent(in) *x*, integer(i4), intent(in) *k*, logical, dimension(:), intent(in), optional *mask*) [private]

Definition at line 520 of file mo_corr.f90.

References mo_corr::autocoeffk::autocoeffk_dp().

Here is the call graph for this function:

15.75.2.7 real(sp) function mo_corr::autocoeffk_sp (real(sp), dimension(:), intent(in) *x*, integer(i4), intent(in) *k*, logical, dimension(:), intent(in), optional *mask*) [private]

Definition at line 539 of file mo_corr.f90.

References mo_corr::autocoeffk::autocoeffk_sp().

Here is the call graph for this function:

15.75.2.8 real(dp) function, dimension(size(k)) mo_corr::autocorr_1d_dp (real(dp), dimension(:), intent(in) *x*, integer(i4), dimension(:), intent(in) *k*, logical, dimension(:), intent(in), optional *mask*) [private]

Definition at line 643 of file mo_corr.f90.

15.75.2.9 real(sp) function, dimension(size(k)) mo_corr::autocorr_1d_sp (real(sp), dimension(:), intent(in) *x*, integer(i4), dimension(:), intent(in) *k*, logical, dimension(:), intent(in), optional *mask*) [private]

Definition at line 669 of file mo_corr.f90.

15.75.2.10 real(dp) function mo_corr::autocorr_dp (real(dp), dimension(:), intent(in) *x*, integer(i4), intent(in) *k*, logical, dimension(:), intent(in), optional *mask*) [private]

Definition at line 606 of file mo_corr.f90.

References mo_corr::autocorr::autocorr_dp().

Here is the call graph for this function:

15.75.2.11 real(sp) function mo_corr::autocorr_sp (real(sp), dimension(:), intent(in) *x*, integer(i4), intent(in) *k*, logical, dimension(:), intent(in), optional *mask*) [private]

Definition at line 625 of file mo_corr.f90.

References mo_corr::autocorr::autocorr_sp().

Here is the call graph for this function:

15.75.2.12 real(dp) function, dimension(size(data1)) mo_corr::corr_dp (real(dp), dimension(:), intent(in) *data1,* real(dp), dimension(:), intent(in) *data2,* integer(i4), intent(out), optional *nadjust,* integer(i4), intent(in), optional *nhigh,* integer(i4), intent(in), optional *nwin*) [private]

Definition at line 697 of file mo_corr.f90.

References mo_corr::corr::corr_dp().

Here is the call graph for this function:

15.75.2.13 real(sp) function, dimension(size(data1)) mo_corr::corr_sp (real(sp), dimension(:), intent(in) *data1,* real(sp), dimension(:), intent(in) *data2,* integer(i4), intent(out), optional *nadjust,* integer(i4), intent(in), optional *nhigh,* integer(i4), intent(in), optional *nwin*) [private]

Definition at line 806 of file mo_corr.f90.

References mo_corr::corr::corr_sp().

Here is the call graph for this function:

15.75.2.14 real(dp) function mo_corr::crosscoeffk_dp (real(dp), dimension(:), intent(in) *x*, real(dp), dimension(:), intent(in) *y*, integer(i4), intent(in) *k*, logical, dimension(:), intent(in), optional *mask*) [private]

Definition at line 916 of file mo_corr.f90.

References mo_corr::crosscoeffk::crosscoeffk_dp().

Here is the call graph for this function:

15.75.2.15 real(sp) function mo_corr::crosscoeffk_sp (real(sp), dimension(:), intent(in) *x*, real(sp), dimension(:), intent(in) *y*, integer(i4), intent(in) *k*, logical, dimension(:), intent(in), optional *mask*) [private]

Definition at line 964 of file mo_corr.f90.

References mo_corr::crosscoeffk::crosscoeffk_sp().

Here is the call graph for this function:

15.75.2.16 real(dp) function mo_corr::crosscorr_dp (real(dp), dimension(:), intent(in) *x*, real(dp), dimension(:), intent(in) *y*, integer(i4), intent(in) *k*, logical, dimension(:), intent(in), optional *mask*) [private]

Definition at line 1014 of file mo_corr.f90.

References mo_corr::crosscorr::crosscorr_dp().

Here is the call graph for this function:

15.75.2.17 real(sp) function mo_corr::crosscorr_sp (real(sp), dimension(:), intent(in) *x*, real(sp), dimension(:), intent(in) *y*, integer(i4), intent(in) *k*, logical, dimension(:), intent(in), optional *mask*) [private]

Definition at line 1034 of file mo_corr.f90.

References mo_corr::crosscorr::crosscorr_sp().

Here is the call graph for this function:

15.75.2.18 subroutine mo_corr::four1_dp (complex(dpc), dimension(:), intent(inout) *data*, integer(i4), intent(in) *isign*) [private]

Definition at line 1098 of file mo_corr.f90.

15.75.2.19 subroutine mo_corr::four1_sp (complex(spc), dimension(:), intent(inout) *data*, integer(i4), intent(in) *isign*) [private]

Definition at line 1063 of file mo_corr.f90.

15.75.2.20 subroutine mo_corr::fourrow_dp (complex(dpc), dimension(:, :), intent(inout) *data*, integer(i4), intent(in) *isign*) [private]

Definition at line 1190 of file mo_corr.f90.

15.75.2.21 subroutine mo_corr::fourrow_sp (complex(spc), dimension(:, :), intent(inout) *data*, integer(i4), intent(in) *isign*) [private]

Definition at line 1140 of file mo_corr.f90.

15.75.2.22 subroutine mo_corr::realft_dp (real(dp), dimension(:), intent(inout) *data,* integer(i4), intent(in) *isign,* complex(dpc), dimension(:), optional, target *zdata*) [private]

Definition at line 1251 of file mo_corr.f90.

References zroots_unity_dp().

Here is the call graph for this function:

15.75.2.23 subroutine mo_corr::realft_sp (real(sp), dimension(:), intent(inout) *data,* integer(i4), intent(in) *isign,* complex(spc), dimension(:), optional, target *zdata*) [private]

Definition at line 1311 of file mo_corr.f90.

References zroots_unity_sp().

Here is the call graph for this function:

15.75.2.24 subroutine mo_corr::swap_1d_dpc (complex(dpc), dimension(:), intent(inout) *a,* complex(dpc), dimension(:), intent(inout) *b*) [private]

Definition at line 1593 of file mo_corr.f90.

15.75.2.25 subroutine mo_corr::swap_1d_spc (complex(spc), dimension(:), intent(inout) *a,* complex(spc), dimension(:), intent(inout) *b*) [private]

Definition at line 1539 of file mo_corr.f90.

15.75.2.26 complex(dpc) function, dimension(nn) mo_corr::zroots_unity_dp (integer(i4), intent(in) *n,* integer(i4), intent(in) *nn*) [private]

Definition at line 1606 of file mo_corr.f90.

Referenced by realft_dp().

Here is the caller graph for this function:

15.75.2.27 complex(spc) function, dimension(nn) mo_corr::zroots_unity_sp (integer(i4), intent(in) *n*, integer(i4), intent(in) *nn*) [private]

Definition at line 1627 of file mo_corr.f90.

Referenced by realft_sp().

Here is the caller graph for this function:

15.75.3 Member Data Documentation

15.75.3.1 integer(i4), parameter mo_corr::npar2_arth = 8 [private]

Definition at line 406 of file mo_corr.f90.

15.75.3.2 integer(i4), parameter mo_corr::npar_arth = 16 [private]

Definition at line 406 of file mo_corr.f90.

The documentation for this module was generated from the following file:

- mo_corr.f90

15.76 mo_dds Module Reference

Dynamically Dimensioned Search (DDS)

Public Member Functions

- real(dp) function, dimension(size(pini)),
  public dds (eval, obj_func, pini, prange, r, seed, maxiter, maxit, mask, tmp_file, funcbest, history)

DDS.

- real(dp) function, dimension(size(pini)),
  public mdds (eval, obj_func, pini, prange, seed, maxiter, maxit, mask, tmp_file, funcbest, history)

  MDDS.
- subroutine neigh_value (x_cur, x_min, x_max, r, new_value)

15.76.1 Detailed Description

Dynamically Dimensioned Search (DDS)

This module provides routines for Dynamically Dimensioned Search (DDS) of Tolson and Shoemaker (2007). It searches the minimum or maximum of a user-specified function, using an n-dimensional continuous global optimization algorithm (DDS).

Authors

Original by Bryan Tolson and later modified by Rohini Kumar. Matthias Cuntz and Juliane Mai for the module, MDDS, etc.

Date

Jul 2012

Definition at line 13 of file mo_dds.f90.

15.76.2 Member Function/Subroutine Documentation

15.76.2.1 real(dp) function, dimension(size(pini)), public mo_dds::dds (procedure(eval_interface), intent(in), pointer *eval,* procedure(objective_interface), intent(in), pointer *obj_func,* real(dp), dimension(:), intent(in) *pini,* real(dp), dimension(:, :), intent(in) *prange,* real(dp), intent(in), optional *r,* integer(i8), intent(in), optional *seed,* integer(i8), intent(in), optional *maxiter,* logical, intent(in), optional *maxit,* logical, dimension(:), intent(in), optional *mask,* character(len = ∗), intent(in), optional *tmp_file,* real(dp), intent(out), optional *funcbest,* real(dp), dimension(:), intent(out), optional, allocatable *history*)

DDS.

Searches Minimum or Maximum of a user-specified function using Dynamically Dimensioned Search (DDS).

DDS is an n-dimensional continuous global optimization algorithm. It is coded as a minimizer but one can give maxit=True in a maximization problem, so that the algorithm minimizes the negative of the objective function F=(-1∗F). The function to be minimized is the first argument of DDS and must be defined as

```
function func(p)
  use mo_kind, only: dp
  implicit none
  real(dp), dimension(:), intent(in) :: p
  real(dp) :: func
end function func
```

Parameters

in	*real(dp) :: obj_func(p)*	Function on which to search the minimum

in	*real(dp) :: pini(:)*	inital value of decision variables
in	*real(dp) :: prange(size(pini),2)*	Min/max range of decision variables
in	*real(dp), optional :: r*	DDS perturbation parameter (default: 0.2)
in	*integer(i8), optional :: seed*	User seed to initialise the random number generator (default: None)
in	*integer(i8), optional :: maxiter*	Maximum number of iteration or function evaluation (default: 1000)
in	*logical, optional :: maxit*	Maximization (.True.) or minimization (.False.) of function (default: .False.)
in	*logical, optional :: mask(size(pini))*	parameter to be optimized (true or false) (default: .True.)
in	*character(len=∗), optional :: tmp_file*	file with temporal output
out	*real(dp), optional :: funcbest*	the best value of the function.
out	*real(dp), optional, allocatable :: history(:)*	the history of best function values, history(maxiter)=funcbest allocatable only to be in correspondance with other optimization routines

Returns

real(dp) :: DDS — The parameters of the point which is estimated to minimize the function.

Author

Written original Bryan Tolson - DDS v1.1
Modified Rohini Kumar, Matthias Cuntz, Juliane Mai

Definition at line 141 of file mo_dds.f90.

References neigh_value().

Here is the call graph for this function:

15.76.2.2 **real(dp) function, dimension(size(pini)), public mo_dds::mdds (procedure(eval_interface), intent(in), pointer *eval,* procedure(objective_interface), intent(in), pointer *obj_func,* real(dp), dimension(:), intent(in) *pini,* real(dp), dimension(:, :), intent(in) *prange,* integer(i8), intent(in), optional *seed,* integer(i8), intent(in), optional *maxiter,* logical, intent(in), optional *maxit,* logical, dimension(:), intent(in), optional *mask,* character(len = ∗), intent(in), optional *tmp_file,* real(dp), intent(out), optional *funcbest,* real(dp), dimension(:), intent(out), optional, allocatable *history*)**

MDDS.

Searches Minimum or Maximum of a user-specified function using the Modified Dynamically Dimensioned Search (DDS). DDS is an n-dimensional continuous global optimization algorithm. It is coded as a minimizer but one can give maxit=True in a maximization problem, so that the algorithm minimizes the negative of the objective function F=(-1∗F).

The function to be minimized is the first argument of DDS and must be defined as

```
function func(p)
  use mo_kind, only: dp
  implicit none
  real(dp), dimension(:), intent(in) :: p
  real(dp) :: func
end function func
```

MDDS extents normal DDS by a continuous reduction of the DDS pertubation parameter r from 0.3 to 0.05, and by allowing a larger function value with a certain probablity.

Parameters

in	*real(dp) :: obj_func(p)*	Function on which to search the minimum
in	*real(dp) :: pini(:)*	inital value of decision variables
in	*real(dp) :: prange(size(pini),2)*	Min/max range of decision variables
in	*integer(i8), optional :: seed*	User seed to initialise the random number generator (default: None)
in	*integer(i8), optional :: maxiter*	Maximum number of iteration or function evaluation (default: 1000)
in	*logical, optional :: maxit*	Maximization (.True.) or minimization (.False.) of function (default: .False.)
in	*logical, optional :: mask(size(pini))*	parameter to be optimized (true or false) (default: .True.)
in	*character(len=∗), optional :: tmp_file*	file with temporal output
out	*real(dp), optional :: funcbest*	the best value of the function.
out	*real(dp), optional, allocatable :: history(:)*	the history of best function values,

Returns

real(dp) :: MDDS — The parameters of the point which is estimated to minimize the function.

Author

Written Matthias Cuntz and Juliane Mai

Definition at line 421 of file mo_dds.f90.

References neigh_value().

Here is the call graph for this function:

15.76.2.3 subroutine mo_dds::neigh_value (real(dp), intent(in) *x_cur,* real(dp), intent(in) *x_min,* real(dp), intent(in) *x_max,* real(dp), intent(in) *r,* real(dp), intent(out) *new_value*)

Definition at line 637 of file mo_dds.f90.

Referenced by dds(), and mdds().

Here is the caller graph for this function:

The documentation for this module was generated from the following file:

- mo_dds.f90

15.77 mo_errormeasures Module Reference

Data Types

- interface bias
- interface kge

  Kling-Gupta-Efficiency measure.
- interface kgenocorr

  Kling-Gupta-Efficiency measure without correlation.

- interface lnnse
- interface mae
- interface mse
- interface nse
- interface rmse
- interface sae
- interface sse
- interface wnse

Public Member Functions

- real(dp) function bias_dp_1d (x, y, mask)
- real(sp) function bias_sp_2d (x, y, mask)
- real(dp) function bias_dp_2d (x, y, mask)
- real(sp) function bias_sp_3d (x, y, mask)
- real(dp) function bias_dp_3d (x, y, mask)
- real(sp) function kge_sp_1d (x, y, mask)
- real(sp) function kge_sp_2d (x, y, mask)
- real(sp) function kge_sp_3d (x, y, mask)
- real(dp) function kge_dp_1d (x, y, mask)
- real(dp) function kge_dp_2d (x, y, mask)
- real(dp) function kge_dp_3d (x, y, mask)
- real(sp) function kgenocorr_sp_1d (x, y, mask)
- real(sp) function kgenocorr_sp_2d (x, y, mask)
- real(sp) function kgenocorr_sp_3d (x, y, mask)
- real(dp) function kgenocorr_dp_1d (x, y, mask)
- real(dp) function kgenocorr_dp_2d (x, y, mask)
- real(dp) function kgenocorr_dp_3d (x, y, mask)
- real(sp) function lnnse_sp_1d (x, y, mask)
- real(dp) function lnnse_dp_1d (x, y, mask)
- real(sp) function lnnse_sp_2d (x, y, mask)
- real(dp) function lnnse_dp_2d (x, y, mask)
- real(sp) function lnnse_sp_3d (x, y, mask)
- real(dp) function lnnse_dp_3d (x, y, mask)
- real(sp) function mae_sp_1d (x, y, mask)
- real(dp) function nse_dp_1d (x, y, mask)
- real(sp) function nse_sp_2d (x, y, mask)
- real(dp) function nse_dp_2d (x, y, mask)
- real(sp) function nse_sp_3d (x, y, mask)
- real(dp) function nse_dp_3d (x, y, mask)
- real(sp) function sae_sp_1d (x, y, mask)
- real(dp) function wnse_dp_1d (x, y, mask)
- real(sp) function wnse_sp_2d (x, y, mask)
- real(dp) function wnse_dp_2d (x, y, mask)
- real(sp) function wnse_sp_3d (x, y, mask)
- real(dp) function wnse_dp_3d (x, y, mask)

Private Member Functions

- real(sp) function bias_sp_1d (x, y, mask)
- real(dp) function mae_dp_1d (x, y, mask)
- real(sp) function mae_sp_2d (x, y, mask)
- real(dp) function mae_dp_2d (x, y, mask)
- real(sp) function mae_sp_3d (x, y, mask)
- real(dp) function mae_dp_3d (x, y, mask)
- real(sp) function mse_sp_1d (x, y, mask)
- real(dp) function mse_dp_1d (x, y, mask)
- real(sp) function mse_sp_2d (x, y, mask)
- real(dp) function mse_dp_2d (x, y, mask)
- real(sp) function mse_sp_3d (x, y, mask)
- real(dp) function mse_dp_3d (x, y, mask)
- real(sp) function nse_sp_1d (x, y, mask)
- real(dp) function sae_dp_1d (x, y, mask)
- real(sp) function sae_sp_2d (x, y, mask)
- real(dp) function sae_dp_2d (x, y, mask)
- real(sp) function sae_sp_3d (x, y, mask)
- real(dp) function sae_dp_3d (x, y, mask)
- real(sp) function sse_sp_1d (x, y, mask)
- real(dp) function sse_dp_1d (x, y, mask)
- real(sp) function sse_sp_2d (x, y, mask)
- real(dp) function sse_dp_2d (x, y, mask)
- real(sp) function sse_sp_3d (x, y, mask)
- real(dp) function sse_dp_3d (x, y, mask)
- real(sp) function rmse_sp_1d (x, y, mask)
- real(dp) function rmse_dp_1d (x, y, mask)
- real(sp) function rmse_sp_2d (x, y, mask)
- real(dp) function rmse_dp_2d (x, y, mask)
- real(sp) function rmse_sp_3d (x, y, mask)
- real(dp) function rmse_dp_3d (x, y, mask)
- real(sp) function wnse_sp_1d (x, y, mask)

15.77.1 Detailed Description

Definition at line 1 of file mo_errormeasures.f90.

15.77.2 Member Function/Subroutine Documentation

15.77.2.1 real(dp) function mo_errormeasures::bias_dp_1d (real(dp), dimension(:), intent(in) *x*, real(dp), dimension(:), intent(in) *y*, logical, dimension(:), intent(in), optional *mask*)

Definition at line 819 of file mo_errormeasures.f90.

References mo_errormeasures::bias::bias_dp_1d().

Here is the call graph for this function:

15.77.2.2 real(dp) function mo_errormeasures::bias_dp_2d (real(dp), dimension(:, :), intent(in) *x*, real(dp), dimension(:, :), intent(in) *y*, logical, dimension(:, :), intent(in), optional *mask*)

Definition at line 896 of file mo_errormeasures.f90.

References mo_errormeasures::bias::bias_dp_2d().

Here is the call graph for this function:

15.77.2.3 real(dp) function mo_errormeasures::bias_dp_3d (real(dp), dimension(:, :, :), intent(in) *x*, real(dp), dimension(:, :, :), intent(in) *y*, logical, dimension(:, :, :), intent(in), optional *mask*)

Definition at line 978 of file mo_errormeasures.f90.

References mo_errormeasures::bias::bias_dp_3d().

Here is the call graph for this function:

15.77.2.4 real(sp) function mo_errormeasures::bias_sp_1d (real(sp), dimension(:), intent(in) *x*, real(sp), dimension(:), intent(in) *y*, logical, dimension(:), intent(in), optional *mask*) [private]

Definition at line 782 of file mo_errormeasures.f90.

References mo_errormeasures::bias::bias_sp_1d().

Here is the call graph for this function:

15.77.2.5 real(sp) function mo_errormeasures::bias_sp_2d (real(sp), dimension(:, :), intent(in) *x*, real(sp), dimension(:, :), intent(in) *y*, logical, dimension(:, :), intent(in), optional *mask*)

Definition at line 855 of file mo_errormeasures.f90.

References mo_errormeasures::bias::bias_sp_2d().

Here is the call graph for this function:

15.77.2.6 real(sp) function mo_errormeasures::bias_sp_3d (real(sp), dimension(:, :, :), intent(in) *x*, real(sp), dimension(:, :, :), intent(in) *y*, logical, dimension(:, :, :), intent(in), optional *mask*)

Definition at line 936 of file mo_errormeasures.f90.

References mo_errormeasures::bias::bias_sp_3d().

Here is the call graph for this function:

15.77.2.7 real(dp) function mo_errormeasures::kge_dp_1d (real(dp), dimension(:), intent(in) *x*, real(dp), dimension(:), intent(in) *y*, logical, dimension(:), intent(in), optional *mask*)

Definition at line 1203 of file mo_errormeasures.f90.

References mo_errormeasures::kge::kge_dp_1d().

Here is the call graph for this function:

15.77.2.8 real(dp) function mo_errormeasures::kge_dp_2d (real(dp), dimension(:, :), intent(in) *x*, real(dp), dimension(:, :), intent(in) *y*, logical, dimension(:, :), intent(in), optional *mask*)

Definition at line 1256 of file mo_errormeasures.f90.

References mo_errormeasures::kge::kge_dp_2d().

Here is the call graph for this function:

15.77.2.9 real(dp) function mo_errormeasures::kge_dp_3d (real(dp), dimension(:, :, :), intent(in) *x*, real(dp), dimension(:, :, :), intent(in) *y*, logical, dimension(:, :, :), intent(in), optional *mask*)

Definition at line 1320 of file mo_errormeasures.f90.

References mo_errormeasures::kge::kge_dp_3d().

Here is the call graph for this function:

15.77.2.10 real(sp) function mo_errormeasures::kge_sp_1d (real(sp), dimension(:), intent(in) *x*, real(sp), dimension(:), intent(in) *y*, logical, dimension(:), intent(in), optional *mask*)

Definition at line 1022 of file mo_errormeasures.f90.

References mo_errormeasures::kge::kge_sp_1d().

Here is the call graph for this function:

15.77.2.11 real(sp) function mo_errormeasures::kge_sp_2d (real(sp), dimension(:, :), intent(in) *x*, real(sp), dimension(:, :), intent(in) *y*, logical, dimension(:, :), intent(in), optional *mask*)

Definition at line 1075 of file mo_errormeasures.f90.

References mo_errormeasures::kge::kge_sp_2d().

Here is the call graph for this function:

15.77.2.12 real(sp) function mo_errormeasures::kge_sp_3d (real(sp), dimension(:, :, :), intent(in) *x*, real(sp), dimension(:, :, :), intent(in) *y*, logical, dimension(:, :, :), intent(in), optional *mask*)

Definition at line 1139 of file mo_errormeasures.f90.

References mo_errormeasures::kge::kge_sp_3d().

Here is the call graph for this function:

15.77.2.13 real(dp) function mo_errormeasures::kgenocorr_dp_1d (real(dp), dimension(:), intent(in) *x*, real(dp), dimension(:), intent(in) *y*, logical, dimension(:), intent(in), optional *mask*)

Definition at line 1549 of file mo_errormeasures.f90.

References mo_errormeasures::kgenocorr::kgenocorr_dp_1d().

Here is the call graph for this function:

15.77.2.14 real(dp) function mo_errormeasures::kgenocorr_dp_2d (real(dp), dimension(:, :), intent(in) *x*, real(dp), dimension(:, :), intent(in) *y*, logical, dimension(:, :), intent(in), optional *mask*)

Definition at line 1599 of file mo_errormeasures.f90.

References mo_errormeasures::kgenocorr::kgenocorr_dp_2d().

Here is the call graph for this function:

15.77.2.15 real(dp) function mo_errormeasures::kgenocorr_dp_3d (real(dp), dimension(:, :, :), intent(in) *x*, real(dp), dimension(:, :, :), intent(in) *y*, logical, dimension(:, :, :), intent(in), optional *mask*)

Definition at line 1655 of file mo_errormeasures.f90.

References mo_errormeasures::kgenocorr::kgenocorr_dp_3d().

Here is the call graph for this function:

15.77.2.16 real(sp) function mo_errormeasures::kgenocorr_sp_1d (real(sp), dimension(:), intent(in) *x*, real(sp), dimension(:), intent(in) *y*, logical, dimension(:), intent(in), optional *mask*)

Definition at line 1386 of file mo_errormeasures.f90.

References mo_errormeasures::kgenocorr::kgenocorr_sp_1d().

Here is the call graph for this function:

15.77.2.17 real(sp) function mo_errormeasures::kgenocorr_sp_2d (real(sp), dimension(:, :), intent(in) *x*, real(sp), dimension(:, :), intent(in) *y*, logical, dimension(:, :), intent(in), optional *mask*)

Definition at line 1436 of file mo_errormeasures.f90.

References mo_errormeasures::kgenocorr::kgenocorr_sp_2d().

Here is the call graph for this function:

15.77.2.18 real(sp) function mo_errormeasures::kgenocorr_sp_3d (real(sp), dimension(:, :, :), intent(in) *x*, real(sp), dimension(:, :, :), intent(in) *y*, logical, dimension(:, :, :), intent(in), optional *mask*)

Definition at line 1492 of file mo_errormeasures.f90.

References mo_errormeasures::kgenocorr::kgenocorr_sp_3d().

Here is the call graph for this function:

15.77.2.19 real(dp) function mo_errormeasures::lnnse_dp_1d (real(dp), dimension(:), intent(in) *x*, real(dp), dimension(:), intent(in) *y*, logical, dimension(:), intent(inout), optional *mask*)

Definition at line 1772 of file mo_errormeasures.f90.

References mo_errormeasures::lnnse::lnnse_dp_1d().

Here is the call graph for this function:

15.77.2.20 real(dp) function mo_errormeasures::lnnse_dp_2d (real(dp), dimension(:, :), intent(in) *x*, real(dp), dimension(:, :), intent(in) *y*, logical, dimension(:, :), intent(inout), optional *mask*)

Definition at line 1886 of file mo_errormeasures.f90.

References mo_errormeasures::lnnse::lnnse_dp_2d().

Here is the call graph for this function:

15.77.2.21 real(dp) function mo_errormeasures::lnnse_dp_3d (real(dp), dimension(:, :, :), intent(in) *x*, real(dp), dimension(:, :, :), intent(in) *y*, logical, dimension(:, :, :), intent(inout), optional *mask*)

Definition at line 2000 of file mo_errormeasures.f90.

References mo_errormeasures::lnnse::lnnse_dp_3d().

Here is the call graph for this function:

15.77.2.22 real(sp) function mo_errormeasures::lnnse_sp_1d (real(sp), dimension(:), intent(in) *x*, real(sp), dimension(:), intent(in) *y*, logical, dimension(:), intent(inout), optional *mask*)

Definition at line 1715 of file mo_errormeasures.f90.

References mo_errormeasures::lnnse::lnnse_sp_1d().

Here is the call graph for this function:

15.77.2.23 real(sp) function mo_errormeasures::lnnse_sp_2d (real(sp), dimension(:, :), intent(in) *x*, real(sp), dimension(:, :), intent(in) *y*, logical, dimension(:, :), intent(inout), optional *mask*)

Definition at line 1829 of file mo_errormeasures.f90.

References mo_errormeasures::lnnse::lnnse_sp_2d().

Here is the call graph for this function:

15.77.2.24 real(sp) function mo_errormeasures::lnnse_sp_3d (real(sp), dimension(:, :, :), intent(in) *x*, real(sp), dimension(:, :, :), intent(in) *y*, logical, dimension(:, :, :), intent(inout), optional *mask*)

Definition at line 1943 of file mo_errormeasures.f90.

References mo_errormeasures::lnnse::lnnse_sp_3d().

Here is the call graph for this function:

15.77.2.25 real(dp) function mo_errormeasures::mae_dp_1d (real(dp), dimension(:), intent(in) *x*, real(dp), dimension(:), intent(in) *y*, logical, dimension(:), intent(in), optional *mask*) [private]

Definition at line 2090 of file mo_errormeasures.f90.

References mo_errormeasures::mae::mae_dp_1d(), and mo_errormeasures::sae::sae_dp_1d().

Here is the call graph for this function:

15.77.2.26 real(dp) function mo_errormeasures::mae_dp_2d (real(dp), dimension(:, :), intent(in) *x*, real(dp), dimension(:, :), intent(in) *y*, logical, dimension(:, :), intent(in), optional *mask*) [private]

Definition at line 2156 of file mo_errormeasures.f90.

References mo_errormeasures::mae::mae_dp_2d(), and mo_errormeasures::sae::sae_dp_2d().

Here is the call graph for this function:

15.77.2.27 real(dp) function mo_errormeasures::mae_dp_3d (real(dp), dimension(:, :, :), intent(in) *x*, real(dp), dimension(:, :, :), intent(in) *y*, logical, dimension(:, :, :), intent(in), optional *mask*) [private]

Definition at line 2223 of file mo_errormeasures.f90.

References mo_errormeasures::mae::mae_dp_3d(), and mo_errormeasures::sae::sae_dp_3d().

Here is the call graph for this function:

15.77.2.28 real(sp) function mo_errormeasures::mae_sp_1d (real(sp), dimension(:), intent(in) *x*, real(sp), dimension(:), intent(in) *y*, logical, dimension(:), intent(in), optional *mask*)

Definition at line 2057 of file mo_errormeasures.f90.

References mo_errormeasures::mae::mae_sp_1d(), and mo_errormeasures::sae::sae_sp_1d().

Here is the call graph for this function:

15.77.2.29 real(sp) function mo_errormeasures::mae_sp_2d (real(sp), dimension(:, :), intent(in) *x*, real(sp), dimension(:, :), intent(in) *y*, logical, dimension(:, :), intent(in), optional *mask*) [private]

Definition at line 2123 of file mo_errormeasures.f90.

References mo_errormeasures::mae::mae_sp_2d(), and mo_errormeasures::sae::sae_sp_2d().

Here is the call graph for this function:

15.77.2.30 real(sp) function mo_errormeasures::mae_sp_3d (real(sp), dimension(:, :, :), intent(in) *x*, real(sp), dimension(:, :, :), intent(in) *y*, logical, dimension(:, :, :), intent(in), optional *mask*) [private]

Definition at line 2189 of file mo_errormeasures.f90.

References mo_errormeasures::mae::mae_sp_3d(), and mo_errormeasures::sae::sae_sp_3d().

Here is the call graph for this function:

15.77.2.31 real(dp) function mo_errormeasures::mse_dp_1d (real(dp), dimension(:), intent(in) *x*, real(dp), dimension(:), intent(in) *y*, logical, dimension(:), intent(in), optional *mask*) [private]

Definition at line 2292 of file mo_errormeasures.f90.

References mo_errormeasures::mse::mse_dp_1d(), and mo_errormeasures::sse::sse_dp_1d().

Here is the call graph for this function:

15.77.2.32 real(dp) function mo_errormeasures::mse_dp_2d (real(dp), dimension(:, :), intent(in) *x*, real(dp), dimension(:, :), intent(in) *y*, logical, dimension(:, :), intent(in), optional *mask*) [private]

Definition at line 2358 of file mo_errormeasures.f90.

References mo_errormeasures::mse::mse_dp_2d(), and mo_errormeasures::sse::sse_dp_2d().

Here is the call graph for this function:

15.77.2.33 real(dp) function mo_errormeasures::mse_dp_3d (real(dp), dimension(:, :, :), intent(in) *x*, real(dp), dimension(:, :, :), intent(in) *y*, logical, dimension(:, :, :), intent(in), optional *mask*) [private]

Definition at line 2425 of file mo_errormeasures.f90.

References mo_errormeasures::mse::mse_dp_3d(), and mo_errormeasures::sse::sse_dp_3d().

Here is the call graph for this function:

15.77.2.34 real(sp) function mo_errormeasures::mse_sp_1d (real(sp), dimension(:), intent(in) *x*, real(sp), dimension(:), intent(in) *y*, logical, dimension(:), intent(in), optional *mask*) [private]

Definition at line 2259 of file mo_errormeasures.f90.

References mo_errormeasures::mse::mse_sp_1d(), and mo_errormeasures::sse::sse_sp_1d().

Here is the call graph for this function:

15.77.2.35 real(sp) function mo_errormeasures::mse_sp_2d (real(sp), dimension(:, :), intent(in) *x*, real(sp), dimension(:, :), intent(in) *y*, logical, dimension(:, :), intent(in), optional *mask*) [private]

Definition at line 2325 of file mo_errormeasures.f90.

References mo_errormeasures::mse::mse_sp_2d(), and mo_errormeasures::sse::sse_sp_2d().

Here is the call graph for this function:

15.77.2.36 real(sp) function mo_errormeasures::mse_sp_3d (real(sp), dimension(:, :, :), intent(in) *x*, real(sp), dimension(:, :, :), intent(in) *y*, logical, dimension(:, :, :), intent(in), optional *mask*) [private]

Definition at line 2391 of file mo_errormeasures.f90.

References mo_errormeasures::mse::mse_sp_3d(), and mo_errormeasures::sse::sse_sp_3d().

Here is the call graph for this function:

15.77.2.37 real(dp) function mo_errormeasures::nse_dp_1d (real(dp), dimension(:), intent(in) *x*, real(dp), dimension(:), intent(in) *y*, logical, dimension(:), intent(in), optional *mask*)

Definition at line 2503 of file mo_errormeasures.f90.

References mo_errormeasures::nse::nse_dp_1d().

Here is the call graph for this function:

15.77.2.38 real(dp) function mo_errormeasures::nse_dp_2d (real(dp), dimension(:, :), intent(in) *x*, real(dp), dimension(:, :), intent(in) *y*, logical, dimension(:, :), intent(in), optional *mask*)

Definition at line 2585 of file mo_errormeasures.f90.

References mo_errormeasures::nse::nse_dp_2d().

Here is the call graph for this function:

15.77.2.39 real(dp) function mo_errormeasures::nse_dp_3d (real(dp), dimension(:, :, :), intent(in) *x*, real(dp), dimension(:, :, :), intent(in) *y*, logical, dimension(:, :, :), intent(in), optional *mask*)

Definition at line 2666 of file mo_errormeasures.f90.

References mo_errormeasures::nse::nse_dp_3d().

Here is the call graph for this function:

15.77.2.40 real(sp) function mo_errormeasures::nse_sp_1d (real(sp), dimension(:), intent(in) *x*, real(sp), dimension(:), intent(in) *y*, logical, dimension(:), intent(in), optional *mask*) [private]

Definition at line 2461 of file mo_errormeasures.f90.

References mo_errormeasures::nse::nse_sp_1d().

Here is the call graph for this function:

15.77.2.41 real(sp) function mo_errormeasures::nse_sp_2d (real(sp), dimension(:, :), intent(in) *x*, real(sp), dimension(:, :), intent(in) *y*, logical, dimension(:, :), intent(in), optional *mask*)

Definition at line 2545 of file mo_errormeasures.f90.

References mo_errormeasures::nse::nse_sp_2d().

Here is the call graph for this function:

15.77.2.42 real(sp) function mo_errormeasures::nse_sp_3d (real(sp), dimension(:, :, :), intent(in) *x*, real(sp), dimension(:, :, :), intent(in) *y*, logical, dimension(:, :, :), intent(in), optional *mask*)

Definition at line 2625 of file mo_errormeasures.f90.

References mo_errormeasures::nse::nse_sp_3d().

Here is the call graph for this function:

15.77.2.43 real(dp) function mo_errormeasures::rmse_dp_1d (real(dp), dimension(:), intent(in) *x*, real(dp), dimension(:), intent(in) *y*, logical, dimension(:), intent(in), optional *mask*) [private]

Definition at line 3174 of file mo_errormeasures.f90.

References mo_errormeasures::mse::mse_dp_1d(), and mo_errormeasures::rmse::rmse_dp_1d().

Here is the call graph for this function:

15.77.2.44 real(dp) function mo_errormeasures::rmse_dp_2d (real(dp), dimension(:, :), intent(in) *x*, real(dp), dimension(:, :), intent(in) *y*, logical, dimension(:, :), intent(in), optional *mask*) [private]

Definition at line 3240 of file mo_errormeasures.f90.

References mo_errormeasures::mse::mse_dp_2d(), and mo_errormeasures::rmse::rmse_dp_2d().

Here is the call graph for this function:

15.77.2.45 real(dp) function mo_errormeasures::rmse_dp_3d (real(dp), dimension(:, :, :), intent(in) *x*, real(dp), dimension(:, :, :), intent(in) *y*, logical, dimension(:, :, :), intent(in), optional *mask*) [private]

Definition at line 3307 of file mo_errormeasures.f90.

References mo_errormeasures::mse::mse_dp_3d(), and mo_errormeasures::rmse::rmse_dp_3d().

Here is the call graph for this function:

15.77.2.46 real(sp) function mo_errormeasures::rmse_sp_1d (real(sp), dimension(:), intent(in) *x*, real(sp), dimension(:), intent(in) *y*, logical, dimension(:), intent(in), optional *mask*) [private]

Definition at line 3141 of file mo_errormeasures.f90.

References mo_errormeasures::mse::mse_sp_1d(), and mo_errormeasures::rmse::rmse_sp_1d().

Here is the call graph for this function:

15.77.2.47 real(sp) function mo_errormeasures::rmse_sp_2d (real(sp), dimension(:, :), intent(in) *x*, real(sp), dimension(:, :), intent(in) *y*, logical, dimension(:, :), intent(in), optional *mask*) [private]

Definition at line 3207 of file mo_errormeasures.f90.

References mo_errormeasures::mse::mse_sp_2d(), and mo_errormeasures::rmse::rmse_sp_2d().

Here is the call graph for this function:

15.77.2.48 real(sp) function mo_errormeasures::rmse_sp_3d (real(sp), dimension(:, :, :), intent(in) *x*, real(sp), dimension(:, :, :), intent(in) *y*, logical, dimension(:, :, :), intent(in), optional *mask*) [private]

Definition at line 3273 of file mo_errormeasures.f90.

References mo_errormeasures::mse::mse_sp_3d(), and mo_errormeasures::rmse::rmse_sp_3d().

Here is the call graph for this function:

15.77.2.49 real(dp) function mo_errormeasures::sae_dp_1d (real(dp), dimension(:), intent(in) *x*, real(dp), dimension(:), intent(in) *y*, logical, dimension(:), intent(in), optional *mask*) [private]

Definition at line 2744 of file mo_errormeasures.f90.

References mo_errormeasures::sae::sae_dp_1d().

Here is the call graph for this function:

15.77.2.50 real(dp) function mo_errormeasures::sae_dp_2d (real(dp), dimension(:, :), intent(in) *x*, real(dp), dimension(:, :), intent(in) *y*, logical, dimension(:, :), intent(in), optional *mask*) [private]

Definition at line 2814 of file mo_errormeasures.f90.

References mo_errormeasures::sae::sae_dp_1d(), and mo_errormeasures::sae::sae_dp_2d().

Here is the call graph for this function:

15.77.2.51 real(dp) function mo_errormeasures::sae_dp_3d (real(dp), dimension(:, :, :), intent(in) *x*, real(dp), dimension(:, :, :), intent(in) *y*, logical, dimension(:, :, :), intent(in), optional *mask*) [private]

Definition at line 2888 of file mo_errormeasures.f90.

References mo_errormeasures::sae::sae_dp_1d(), and mo_errormeasures::sae::sae_dp_3d().

Here is the call graph for this function:

15.77.2.52 real(sp) function mo_errormeasures::sae_sp_1d (real(sp), dimension(:), intent(in) *x*, real(sp), dimension(:), intent(in) *y*, logical, dimension(:), intent(in), optional *mask*)

Definition at line 2710 of file mo_errormeasures.f90.

References mo_errormeasures::sae::sae_sp_1d().

Here is the call graph for this function:

15.77.2.53 real(sp) function mo_errormeasures::sae_sp_2d (real(sp), dimension(:, :), intent(in) *x*, real(sp), dimension(:, :), intent(in) *y*, logical, dimension(:, :), intent(in), optional *mask*) [private]

Definition at line 2778 of file mo_errormeasures.f90.

References mo_errormeasures::sae::sae_sp_1d(), and mo_errormeasures::sae::sae_sp_2d().

Here is the call graph for this function:

15.77.2.54 real(sp) function mo_errormeasures::sae_sp_3d (real(sp), dimension(:, :, :), intent(in) *x*, real(sp), dimension(:, :, :), intent(in) *y*, logical, dimension(:, :, :), intent(in), optional *mask*) [private]

Definition at line 2850 of file mo_errormeasures.f90.

References mo_errormeasures::sae::sae_sp_1d(), and mo_errormeasures::sae::sae_sp_3d().

Here is the call graph for this function:

15.77.2.55 real(dp) function mo_errormeasures::sse_dp_1d (real(dp), dimension(:), intent(in) *x*, real(dp), dimension(:), intent(in) *y*, logical, dimension(:), intent(in), optional *mask*) [private]

Definition at line 2962 of file mo_errormeasures.f90.

References mo_errormeasures::sse::sse_dp_1d().

Here is the call graph for this function:

15.77.2.56 real(dp) function mo_errormeasures::sse_dp_2d (real(dp), dimension(:, :), intent(in) *x*, real(dp), dimension(:, :), intent(in) *y*, logical, dimension(:, :), intent(in), optional *mask*) [private]

Definition at line 3030 of file mo_errormeasures.f90.

References mo_errormeasures::sse::sse_dp_1d(), and mo_errormeasures::sse::sse_dp_2d().

Here is the call graph for this function:

15.77.2.57 real(dp) function mo_errormeasures::sse_dp_3d (real(dp), dimension(:, :, :), intent(in) *x*, real(dp), dimension(:, :, :), intent(in) *y*, logical, dimension(:, :, :), intent(in), optional *mask*) [private]

Definition at line 3102 of file mo_errormeasures.f90.

References mo_errormeasures::sse::sse_dp_1d(), and mo_errormeasures::sse::sse_dp_3d().

Here is the call graph for this function:

15.77.2.58 real(sp) function mo_errormeasures::sse_sp_1d (real(sp), dimension(:), intent(in) *x*, real(sp), dimension(:), intent(in) *y*, logical, dimension(:), intent(in), optional *mask*) [private]

Definition at line 2928 of file mo_errormeasures.f90.

References mo_errormeasures::sse::sse_sp_1d().

Here is the call graph for this function:

15.77.2.59 real(sp) function mo_errormeasures::sse_sp_2d (real(sp), dimension(:, :), intent(in) *x*, real(sp), dimension(:, :), intent(in) *y*, logical, dimension(:, :), intent(in), optional *mask*) [private]

Definition at line 2995 of file mo_errormeasures.f90.

References mo_errormeasures::sse::sse_sp_1d(), and mo_errormeasures::sse::sse_sp_2d().

Here is the call graph for this function:

15.77.2.60 real(sp) function mo_errormeasures::sse_sp_3d (real(sp), dimension(:, :, :), intent(in) *x*, real(sp), dimension(:, :, :), intent(in) *y*, logical, dimension(:, :, :), intent(in), optional *mask*) [private]

Definition at line 3065 of file mo_errormeasures.f90.

References mo_errormeasures::sse::sse_sp_1d(), and mo_errormeasures::sse::sse_sp_3d().

Here is the call graph for this function:

15.77.2.61 real(dp) function mo_errormeasures::wnse_dp_1d (real(dp), dimension(:), intent(in) *x*, real(dp), dimension(:), intent(in) *y*, logical, dimension(:), intent(in), optional *mask*)

Definition at line 3386 of file mo_errormeasures.f90.

References mo_errormeasures::wnse::wnse_dp_1d().

Here is the call graph for this function:

15.77.2.62 real(dp) function mo_errormeasures::wnse_dp_2d (real(dp), dimension(:,:), intent(in) *x*, real(dp), dimension(:,:), intent(in) *y*, logical, dimension(:,:), intent(in), optional *mask*)

Definition at line 3469 of file mo_errormeasures.f90.

References mo_errormeasures::wnse::wnse_dp_2d().

Here is the call graph for this function:

15.77.2.63 real(dp) function mo_errormeasures::wnse_dp_3d (real(dp), dimension(:,:,:), intent(in) *x*, real(dp), dimension(:,:,:), intent(in) *y*, logical, dimension(:,:,:), intent(in), optional *mask*)

Definition at line 3550 of file mo_errormeasures.f90.

References mo_errormeasures::wnse::wnse_dp_3d().

Here is the call graph for this function:

15.77.2.64 real(sp) function mo_errormeasures::wnse_sp_1d (real(sp), dimension(:), intent(in) *x*, real(sp), dimension(:), intent(in) *y*, logical, dimension(:), intent(in), optional *mask*) [private]

Definition at line 3343 of file mo_errormeasures.f90.

References mo_errormeasures::wnse::wnse_sp_1d().

Here is the call graph for this function:

15.77.2.65 real(sp) function mo_errormeasures::wnse_sp_2d (real(sp), dimension(:,:), intent(in) *x*, real(sp), dimension(:,:), intent(in) *y*, logical, dimension(:,:), intent(in), optional *mask*)

Definition at line 3429 of file mo_errormeasures.f90.

References mo_errormeasures::wnse::wnse_sp_2d().

Here is the call graph for this function:

15.77.2.66 real(sp) function mo_errormeasures::wnse_sp_3d (real(sp), dimension(:,:,:), intent(in) *x*, real(sp), dimension(:,:,:), intent(in) *y*, logical, dimension(:,:,:), intent(in), optional *mask*)

Definition at line 3509 of file mo_errormeasures.f90.

References mo_errormeasures::wnse::wnse_sp_3d().

Here is the call graph for this function:

The documentation for this module was generated from the following file:

- mo_errormeasures.f90

15.78 mo_file Module Reference

Provides file names and units for mHM.

Collaboration diagram for mo_file:

Public Attributes

- character(len=∗), parameter version = '5.9'

  Current mHM model version.

- character(len=∗), parameter version_date = 'June 2018'

  Time of current mHM model version release.
- character(len=∗), parameter file_main = 'mhm_driver.f90'

  Driver file.
- character(len=∗), parameter file_namelist_mhm = 'mhm.nml'

  Namelist file name.
- integer, parameter unamelist_mhm = 30

  Unit for namelist.
- character(len=∗), parameter file_namelist_mhm_param = 'mhm_parameter.nml'

  Parameter namelists file name.
- integer, parameter unamelist_mhm_param = 31

  Unit for namelist.
- character(len=∗), parameter file_defoutput = 'mhm_outputs.nml'

  file defining mHM's outputs
- integer, parameter udefoutput = 67

  Unit for file defining mHM's outputs.
- integer, parameter utws = 77

  unit for tws time series

15.78.1 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

Definition at line 14 of file mo_file.f90.

15.78.2 Member Data Documentation

15.78.2.1 character(len = ∗), parameter mo_file::file_defoutput = 'mhm_outputs.nml'

file defining mHM's outputs

Definition at line 34 of file mo_file.f90.

15.78.2.2 character(len = ∗), parameter mo_file::file_main = 'mhm_driver.f90'

Driver file.

Definition at line 23 of file mo_file.f90.

15.78.2.3 character(len = ∗), parameter mo_file::file_namelist_mhm = 'mhm.nml'

Namelist file name.

Definition at line 25 of file mo_file.f90.

15.78.2.4 character(len = ∗), parameter mo_file::file_namelist_mhm_param = 'mhm_parameter.nml'

Parameter namelists file name.

Definition at line 29 of file mo_file.f90.

15.78.2.5 integer, parameter mo_file::udefoutput = 67

Unit for file defining mHM's outputs.

Definition at line 36 of file mo_file.f90.

15.78.2.6 integer, parameter mo_file::unamelist_mhm = 30

Unit for namelist.

Definition at line 27 of file mo_file.f90.

15.78.2.7 integer, parameter mo_file::unamelist_mhm_param = 31

Unit for namelist.

Definition at line 31 of file mo_file.f90.

15.78.2.8 integer, parameter mo_file::utws = 77

unit for tws time series

Definition at line 39 of file mo_file.f90.

15.78.2.9 character(len = ∗), parameter mo_file::version = '5.9'

Current mHM model version.

Definition at line 19 of file mo_file.f90.

15.78.2.10 character(len = ∗), parameter mo_file::version_date = 'June 2018'

Time of current mHM model version release.

Definition at line 21 of file mo_file.f90.

The documentation for this module was generated from the following file:

- mo_file.f90

15.79 mo_finish Module Reference

Finish a program gracefully.

Public Member Functions

- subroutine, public finish (name, text, unit)

  Finish a program gracefully.

15.79.1 Detailed Description

Finish a program gracefully.

This module supplies a routine that writes out final comments and then stops.

Authors

Original of Echam5, (C) MPI-MET, Hamburg, Germany.
Modified Matthias Cuntz

Date

Jan 2011

Definition at line 11 of file mo_finish.f90.

15.79.2 Member Function/Subroutine Documentation

15.79.2.1 subroutine, public mo_finish::finish (character(len = ∗), intent(in) *name,* character(len = ∗), intent(in), optional *text,* integer, intent(in), optional *unit*)

Finish a program gracefully.

Stop a program but writing out a message first that is separated from earlier output by ----------- (i.e. the separator of mo_string_utils)

Parameters

in	*character(len=∗) :: name*	First string separated from otional second by :
in	*character(len=∗), optional :: text*	Second string separated by :
in	*integer, optional :: unit*	File unit for write (default: ∗)

Author

Written, Matthias Cuntz - modified from Echam5, (C) MPI-MET, Hamburg, Germany

Date

Dec 2011

Definition at line 96 of file mo_finish.f90.

Referenced by mo_nml::close_nml(), mhm_driver(), mrm_driver(), mo_nml::open_nml(), and mo_nml::position_↩nml().

Here is the caller graph for this function:

The documentation for this module was generated from the following file:

- mo_finish.f90

15.80 mo_global_variables Module Reference

Global variables ONLY used in reading, writing and startup.

Collaboration diagram for mo_global_variables:

Data Types

- type twsstructure

Public Attributes

- integer(i4) timestep_model_outputs
- logical, dimension(noutflxstate) outputflxstate
- integer(i4), dimension(:),
  allocatable, public timestep_model_inputs
- logical, public read_meteo_weights
- character(256), public inputformat_meteo_forcings
- integer(i4), public timestep_sm_input
- integer(i4), public timestep_neutrons_input
- integer(i4), public timestep_et_input
- character(256), dimension(:),
  allocatable, public dirprecipitation

- character(256), dimension(:), allocatable, public dirtemperature
- character(256), dimension(:), allocatable, public dirmintemperature
- character(256), dimension(:), allocatable, public dirmaxtemperature
- character(256), dimension(:), allocatable, public dirnetradiation
- character(256), dimension(:), allocatable, public dirabsvappressure
- character(256), dimension(:), allocatable, public dirwindspeed
- character(256), dimension(:), allocatable, public dirreferenceet
- character(256), dimension(:), allocatable, public dirsoil_moisture
- character(256), dimension(:), allocatable, public filetws
- character(256), dimension(:), allocatable, public dirneutrons
- character(256), dimension(:), allocatable, public direvapotranspiration
- integer(i4), parameter, public routingstates = 2
- type(twsstructure), public basin_avg_tws_obs
- real(dp), dimension(:,:), allocatable, public basin_avg_tws_sim
- integer(i4), public nmeasperday_tws
- type(grid), dimension(:), allocatable, public level2
- real(dp), dimension(:,:,:), allocatable, public l1_temp_weights
- real(dp), dimension(:,:,:), allocatable, public l1_pet_weights
- real(dp), dimension(:,:,:), allocatable, public l1_pre_weights
- real(dp), dimension(:,:), allocatable, public l1_pre
- real(dp), dimension(:,:), allocatable, public l1_temp
- real(dp), dimension(:,:), allocatable, public l1_pet
- real(dp), dimension(:,:), allocatable, public l1_tmin
- real(dp), dimension(:,:), allocatable, public l1_tmax
- real(dp), dimension(:,:), allocatable, public l1_netrad
- real(dp), dimension(:,:), allocatable, public l1_absvappress
- real(dp), dimension(:,:), allocatable, public l1_windspeed
- real(dp), dimension(:,:), allocatable, public l1_sm
- logical, dimension(:,:), allocatable, public l1_sm_mask
- integer(i4) ntimesteps_l1_sm

- integer(i4) nsoilhorizons_sm_input
- real(dp), dimension(:,:), allocatable, public l1_neutronsdata
- logical, dimension(:,:), allocatable, public l1_neutronsdata_mask
- integer(i4) ntimesteps_l1_neutrons
- real(dp), dimension(:,:), allocatable, public l1_et
- logical, dimension(:,:), allocatable, public l1_et_mask
- integer(i4) ntimesteps_l1_et
- real(dp), dimension(:), allocatable, public l1_inter
- real(dp), dimension(:), allocatable, public l1_snowpack
- real(dp), dimension(:), allocatable, public l1_sealstw
- real(dp), dimension(:,:), allocatable, public l1_soilmoist
- real(dp), dimension(:), allocatable, public l1_unsatstw
- real(dp), dimension(:), allocatable, public l1_satstw
- real(dp), dimension(:), allocatable, public l1_neutrons
- real(dp), dimension(:), allocatable, public l1_pet_calc
- real(dp), dimension(:,:), allocatable, public l1_aetsoil
- real(dp), dimension(:), allocatable, public l1_aetcanopy
- real(dp), dimension(:), allocatable, public l1_aetsealed
- real(dp), dimension(:), allocatable, public l1_baseflow
- real(dp), dimension(:,:), allocatable, public l1_infilsoil
- real(dp), dimension(:), allocatable, public l1_fastrunoff
- real(dp), dimension(:), allocatable, public l1_melt
- real(dp), dimension(:), allocatable, public l1_percol
- real(dp), dimension(:), allocatable, public l1_preeffect
- real(dp), dimension(:), allocatable, public l1_rain
- real(dp), dimension(:), allocatable, public l1_runoffseal
- real(dp), dimension(:), allocatable, public l1_slowrunoff
- real(dp), dimension(:), allocatable, public l1_snow
- real(dp), dimension(:), allocatable, public l1_throughfall
- real(dp), dimension(:), allocatable, public l1_total_runoff

- real(dp), dimension(int(yearmonths, i4)), public evap_coeff
- real(dp), dimension(int(yearmonths, i4)), public fday_prec
- real(dp), dimension(int(yearmonths, i4)), public fnight_prec
- real(dp), dimension(int(yearmonths, i4)), public fday_pet
- real(dp), dimension(int(yearmonths, i4)), public fnight_pet
- real(dp), dimension(int(yearmonths, i4)), public fday_temp
- real(dp), dimension(int(yearmonths, i4)), public fnight_temp
- real(dp), dimension(:), allocatable, public neutron_integral_afast

15.80.1 Detailed Description

Global variables ONLY used in reading, writing and startup.

TODO: add description

Authors

Luis Samaniego

Date

Dec 2012

Definition at line 15 of file mo_global_variables.f90.

15.80.2 Member Data Documentation

15.80.2.1 type(twsstructure), public mo_global_variables::basin_avg_tws_obs

Definition at line 97 of file mo_global_variables.f90.

15.80.2.2 real(dp), dimension(:, :), allocatable, public mo_global_variables::basin_avg_tws_sim

Definition at line 99 of file mo_global_variables.f90.

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

Definition at line 76 of file mo_global_variables.f90.

15.80.2.4 character(256), dimension(:), allocatable, public mo_global_variables::direvapotranspiration

Definition at line 82 of file mo_global_variables.f90.

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

Definition at line 74 of file mo_global_variables.f90.

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

Definition at line 73 of file mo_global_variables.f90.

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

Definition at line 75 of file mo_global_variables.f90.

15.80.2.8 character(256), dimension(:), allocatable, public mo_global_variables::dirneutrons

Definition at line 81 of file mo_global_variables.f90.

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

Definition at line 71 of file mo_global_variables.f90.

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

Definition at line 78 of file mo_global_variables.f90.

15.80.2.11 character(256), dimension(:), allocatable, public mo_global_variables::dirsoil_moisture

Definition at line 79 of file mo_global_variables.f90.

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

Definition at line 72 of file mo_global_variables.f90.

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

Definition at line 77 of file mo_global_variables.f90.

15.80.2.14 real(dp), dimension(int(yearmonths, i4)), public mo_global_variables::evap_coeff

Definition at line 179 of file mo_global_variables.f90.

15.80.2.15 real(dp), dimension(int(yearmonths, i4)), public mo_global_variables::fday_pet

Definition at line 182 of file mo_global_variables.f90.

15.80.2.16 real(dp), dimension(int(yearmonths, i4)), public mo_global_variables::fday_prec

Definition at line 180 of file mo_global_variables.f90.

15.80.2.17 real(dp), dimension(int(yearmonths, i4)), public mo_global_variables::fday_temp

Definition at line 184 of file mo_global_variables.f90.

15.80.2.18 character(256), dimension(:), allocatable, public mo_global_variables::filetws

Definition at line 80 of file mo_global_variables.f90.

15.80.2.19 real(dp), dimension(int(yearmonths, i4)), public mo_global_variables::fnight_pet

Definition at line 183 of file mo_global_variables.f90.

15.80.2.20 real(dp), dimension(int(yearmonths, i4)), public mo_global_variables::fnight_prec

Definition at line 181 of file mo_global_variables.f90.

15.80.2.21 real(dp), dimension(int(yearmonths, i4)), public mo_global_variables::fnight_temp

Definition at line 185 of file mo_global_variables.f90.

15.80.2.22 character(256), public mo_global_variables::inputformat_meteo_forcings

Definition at line 62 of file mo_global_variables.f90.

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

Definition at line 123 of file mo_global_variables.f90.

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

Definition at line 159 of file mo_global_variables.f90.

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

Definition at line 160 of file mo_global_variables.f90.

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

Definition at line 158 of file mo_global_variables.f90.

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

Definition at line 161 of file mo_global_variables.f90.

15.80.2.28 real(dp), dimension(:, :), allocatable, public mo_global_variables::l1_et

Definition at line 139 of file mo_global_variables.f90.

15.80.2.29 logical, dimension(:, :), allocatable, public mo_global_variables::l1_et_mask

Definition at line 140 of file mo_global_variables.f90.

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

Definition at line 163 of file mo_global_variables.f90.

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

Definition at line 162 of file mo_global_variables.f90.

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

Definition at line 146 of file mo_global_variables.f90.

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

Definition at line 164 of file mo_global_variables.f90.

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

Definition at line 122 of file mo_global_variables.f90.

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

Definition at line 152 of file mo_global_variables.f90.

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

Definition at line 135 of file mo_global_variables.f90.

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

Definition at line 136 of file mo_global_variables.f90.

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

Definition at line 165 of file mo_global_variables.f90.

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

Definition at line 119 of file mo_global_variables.f90.

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

Definition at line 157 of file mo_global_variables.f90.

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

Definition at line 115 of file mo_global_variables.f90.

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

Definition at line 117 of file mo_global_variables.f90.

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

Definition at line 116 of file mo_global_variables.f90.

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

Definition at line 166 of file mo_global_variables.f90.

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

Definition at line 167 of file mo_global_variables.f90.

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

Definition at line 168 of file mo_global_variables.f90.

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

Definition at line 151 of file mo_global_variables.f90.

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

Definition at line 148 of file mo_global_variables.f90.

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

Definition at line 169 of file mo_global_variables.f90.

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

Definition at line 130 of file mo_global_variables.f90.

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

Definition at line 131 of file mo_global_variables.f90.

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

Definition at line 170 of file mo_global_variables.f90.

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

Definition at line 147 of file mo_global_variables.f90.

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

Definition at line 149 of file mo_global_variables.f90.

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

Definition at line 118 of file mo_global_variables.f90.

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

Definition at line 114 of file mo_global_variables.f90.

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

Definition at line 171 of file mo_global_variables.f90.

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

Definition at line 121 of file mo_global_variables.f90.

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

Definition at line 120 of file mo_global_variables.f90.

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

Definition at line 172 of file mo_global_variables.f90.

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

Definition at line 150 of file mo_global_variables.f90.

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

Definition at line 124 of file mo_global_variables.f90.

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

Definition at line 106 of file mo_global_variables.f90.

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

Definition at line 192 of file mo_global_variables.f90.

15.80.2.65 integer(i4), public mo_global_variables::nmeasperday_tws

Definition at line 100 of file mo_global_variables.f90.

15.80.2.66 integer(i4) mo_global_variables::nsoilhorizons_sm_input

Definition at line 133 of file mo_global_variables.f90.

15.80.2.67 integer(i4) mo_global_variables::ntimesteps_l1_et

Definition at line 141 of file mo_global_variables.f90.

15.80.2.68 integer(i4) mo_global_variables::ntimesteps_l1_neutrons

Definition at line 137 of file mo_global_variables.f90.

15.80.2.69 integer(i4) mo_global_variables::ntimesteps_l1_sm

Definition at line 132 of file mo_global_variables.f90.

15.80.2.70 logical, dimension(noutflxstate) mo_global_variables::outputflxstate

Definition at line 55 of file mo_global_variables.f90.

15.80.2.71 logical, public mo_global_variables::read_meteo_weights

Definition at line 61 of file mo_global_variables.f90.

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

Definition at line 87 of file mo_global_variables.f90.

15.80.2.73 integer(i4), public mo_global_variables::timestep_et_input

Definition at line 66 of file mo_global_variables.f90.

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

Definition at line 60 of file mo_global_variables.f90.

15.80.2.75 integer(i4) mo_global_variables::timestep_model_outputs

Definition at line 54 of file mo_global_variables.f90.

15.80.2.76 integer(i4), public mo_global_variables::timestep_neutrons_input

Definition at line 65 of file mo_global_variables.f90.

15.80.2.77 integer(i4), public mo_global_variables::timestep_sm_input

Definition at line 64 of file mo_global_variables.f90.

The documentation for this module was generated from the following file:

- mo_global_variables.f90

15.81 mo_grid Module Reference

TODO: add description.

Public Member Functions

- subroutine, public init_lowres_level (highres, target_resolution, lowres, highres_lowres_remap)

  Level-1 variable initialization.
- subroutine, public set_basin_indices (grids)

  TODO: add description.
- subroutine, public l0_grid_setup (new_grid)

  level 0 variable initialization
- subroutine, public mapcoordinates (level, y, x)

  Generate map coordinates.
- subroutine, public geocoordinates (level, lat, lon)

  Generate geographic coordinates.
- subroutine calculate_grid_properties (nrowsIn, ncolsIn, xllcornerIn, yllcornerIn, cellsizeIn, aimingResolution, nrowsOut, ncolsOut, xllcornerOut, yllcornerOut, cellsizeOut)

  Calculates basic grid properties at a required coarser level using information of a given finer level. Calculates basic grid properties at a required coarser level (e.g., L11) using information of a given finer level (e.g., L0). Basic grid properties such as nrows, ncols, xllcorner, yllcorner cellsize are estimated in this routine.

15.81.1 Detailed Description

TODO: add description.

TODO: add description

Authors

Robert Schweppe

Date

Jun 2018

Definition at line 13 of file mo_grid.f90.

15.81.2 Member Function/Subroutine Documentation

15.81.2.1 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) :: aiming↩ Resolution*	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 484 of file mo_grid.f90.

References mo_message::message().

Referenced by init_lowres_level().

Here is the call graph for this function:

Here is the caller graph for this function:

15.81.2.2 subroutine, public mo_grid::geocoordinates (type(grid), intent(in) *level,* real(dp), dimension(:, :), intent(out), allocatable *lat,* real(dp), dimension(:, :), intent(out), allocatable *lon*)

Generate geographic coordinates.

Generate geographic coordinate arrays for given basin and level

Parameters

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

Authors

Matthias Zink

Date

Apr 2013

Definition at line 428 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:

15.81.2.3 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(Grid↩ Remapper), optional :: highres_lowres↩ _remap*	

Authors

Rohini Kumar

Date

Jan 2013

Definition at line 59 of file mo_grid.f90.

References calculate_grid_properties().

Referenced by mo_startup::l2_variable_init(), and mo_mrm_init::mrm_init().

Here is the call graph for this function:

Here is the caller graph for this function:

15.81.2.4 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 263 of file mo_grid.f90.

Referenced by mo_mrm_init::mrm_init().

Here is the caller graph for this function:

15.81.2.5 subroutine, public mo_grid::mapcoordinates (type(grid), intent(in) *level*, real(dp), dimension(:), intent(out), allocatable *y*, real(dp), dimension(:), intent(out), allocatable *x*)

Generate map coordinates.

Generate map coordinate arrays for given basin and level

Parameters

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

Authors

Matthias Zink

Date

Apr 2013

Definition at line 365 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:

15.81.2.6 subroutine, public mo_grid::set_basin_indices (type(grid), dimension(:), intent(inout) *grids*)

TODO: add description.

TODO: add description

Parameters

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

Authors

Robert Schweppe

Date

Jun 2018

Definition at line 205 of file mo_grid.f90.

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

Here is the caller graph for this function:

The documentation for this module was generated from the following file:

- mo_grid.f90

15.82 mo_init_states Module Reference

Initialization of all state variables of mHM.

Public Member Functions

- 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.

15.82.1 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

Definition at line 16 of file mo_init_states.f90.

15.82.2 Member Function/Subroutine Documentation

15.82.2.1 subroutine, public mo_init_states::variables_alloc (integer(i4), intent(in) *ncells1*)

Allocation of space for mHM related L1 and L11 variables.

Allocation of space for mHM related L1 and L11 variables (e.g., states, fluxes, and parameters) for a given basin. Variables allocated here is defined in them mo_global_variables.f90 file. After allocating any variable in this routine, initalize them in the following variables_default_init subroutine:

Parameters

in	*integer(i4) :: ncells1*	

Authors

Rohini Kumar

Date

Jan 2013

Definition at line 62 of file mo_init_states.f90.

References mo_runoff::l1_total_runoff().

Referenced by mo_startup::mhm_initialize().

Here is the call graph for this function:

Here is the caller graph for this function:

15.82.2.2 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_runoff::l1_total_runoff().

Referenced by mo_mhm_eval::mhm_eval().

Here is the call graph for this function:

Here is the caller graph for this function:

The documentation for this module was generated from the following file:

- mo_init_states.f90

15.83 mo_julian Module Reference

Julian date conversion routines.

Collaboration diagram for mo_julian:

Data Types

- interface setcalendar

Public Member Functions

- elemental subroutine, public caldat (julian, dd, mm, yy, calendar)

  Day, month and year from Julian day in the current or given calendar.
- elemental subroutine, public dec2date (julian, dd, mm, yy, hh, nn, ss, calendar)

  Day, month, year, hour, minute, and second from fractional Julian day in the current or given calendar.
- elemental real(dp) function, public date2dec (dd, mm, yy, hh, nn, ss, calendar)

  Fractional Julian day from day, month, year, hour, minute, second in the current calendar.
- elemental integer(i4) function,
  public julday (dd, mm, yy, calendar)

  Julian day from day, month and year in the current or given calendar.
- elemental subroutine, public caldatjulian (julian, dd, mm, yy)

  Day, month and year from Julian day.
- elemental integer(i4) function,
  public ndays (dd, mm, yy)

  IMSL Julian day from day, month and year.
- elemental subroutine, public ndyin (julian, dd, mm, yy)

  Day, month and year from IMSL Julian day.

Private Member Functions

- subroutine setcalendarstring (selector)

  Set module private variable calendar.
- subroutine setcalendarinteger (selector)

  Set module private variable calendar.
- pure integer(i4) function selectcalendar (selector)

  Select a calendar.
- elemental real(dp) function date2decjulian (dd, mm, yy, hh, nn, ss)

  Fractional Julian day from day, month, year, hour, minute, second.
- elemental subroutine dec2datejulian (julian, dd, mm, yy, hh, nn, ss)

Day, month, year, hour, minute, and second from fractional Julian day.

- elemental integer(i4) function juldayjulian (dd, mm, yy)

  Julian day from day, month and year.
- elemental subroutine caldat360 (julian, dd, mm, yy)

  Day, month and year from Julian day in a 360 day calendar.
- elemental integer(i4) function julday360 (dd, mm, yy)

  Julian day from day, month and year in a 360_day calendar.
- elemental subroutine dec2date360 (julian, dd, mm, yy, hh, nn, ss)

  Day, month, year, hour, minute, and second from fractional Julian day in a 360_day calendar.
- elemental real(dp) function date2dec360 (dd, mm, yy, hh, nn, ss)

  Fractional Julian day from day, month, year, hour, minute, second in 360 day calendar.
- elemental subroutine caldat365 (julian, dd, mm, yy)

  Day, month and year from Julian day in a 365 day calendar.
- elemental integer(i4) function julday365 (dd, mm, yy)

  Julian day from day, month and year in a 365_day calendar.
- elemental subroutine dec2date365 (julian, dd, mm, yy, hh, nn, ss)

  Day, month, year, hour, minute, and second from fractional Julian day in a 365_day calendar.
- elemental real(dp) function date2dec365 (dd, mm, yy, hh, nn, ss)

  Fractional Julian day from day, month, year, hour, minute, second in 365 day calendar.

Private Attributes

- integer(i4), save, private calendar = 1

15.83.1 Detailed Description

Julian date conversion routines.

Julian date to and from day, month, year, and also from day, month, year, hour, minute, and second. Also convience routines for Julian dates of IMSL are provided.

Note

Julian day definition starts at noon of the 1st January 4713 BC.
Here, the astronomical definition is used, i.e. the year 1 BC (historic) is counted as 0 (astronomic), 2 BC is -1, etc.
This means that Julian day definition starts as 01.01.-4712 in astronomical units.

julday and caldat start at midnight of the 1st January 4713 BC. So date2dec and julday as well as dec2date and caldat are shifted by half a day.
Use date2dec with dec2date together for fractional Julian dates and use julday with caldat together for integer Julian days.

Author

Matthias Cuntz

Date

Dec 2011

Definition at line 21 of file mo_julian.f90.

15.83.2 Member Function/Subroutine Documentation

15.83.2.1 elemental subroutine, public mo_julian::caldat (integer(i4), intent(in) *julian,* integer(i4), intent(out) *dd,* integer(i4), intent(out) *mm,* integer(i4), intent(out) *yy,* integer(i4), intent(in), optional *calendar*)

Day, month and year from Julian day in the current or given calendar.

Wrapper around the calendar specific caldat procedures. Inverse of the function julday. Here julian is input as a Julian Day Number, and the routine outputs d0d, mm, and yy as the day, month, and year on which the specified Julian Day started at noon. The zeroth Julian Day depends on the called procedure. See their documentation for details.

Parameters

in	*integer(i4) :: julday*	Julian day
out	*integer(i4) :: dd*	Day in month of Julian day
out	*integer(i4) :: mm*	Month in year of Julian day
out	*integer(i4) :: yy*	Year of Julian day
in	*integer(i4) :: calendar*	The calendar to use, the global calendar will be used by default

Author

Written, David Schaefer

Date

Jan 2015

Definition at line 262 of file mo_julian.f90.

References caldat360(), caldat365(), caldatjulian(), and selectcalendar().

Referenced by mo_meteo_forcings::chunk_size(), dec2datejulian(), mo_read_forcing_nc::get_time_vector_and_select(), mo_meteo_forcings::is_read(), mo_mhm_eval::mhm_eval(), mo_mrm_eval::mrm_eval(), mo_mrm_write::mrm_write_output_fluxes(), ndyin(), mo_objective_function::objective_kge_q_rmse_et(), and mo_objective_function::objective_kge_q_rmse_tws().

Here is the call graph for this function:

Here is the caller graph for this function:

15.83.2.2 elemental subroutine mo_julian::caldat360 (integer(i4), intent(in) *julian,* integer(i4), intent(out) *dd,* integer(i4), intent(out) *mm,* integer(i4), intent(out) *yy*) [private]

Day, month and year from Julian day in a 360 day calendar.

Inverse of the function julday360. Here julian is input as a Julian Day Number, and the routine outputs dd, mm, and yy as the day, month, and year on which the specified Julian Day started at noon. The zeroth Julian Day here is 01.01.0000

Parameters

in	*integer(i4) :: julday*	Julian day
out	*integer(i4) :: dd*	Day in month of Julian day
out	*integer(i4) :: mm*	Month in year of Julian day
out	*integer(i4) :: yy*	Year of Julian day

Author

Written, David Schaefer

Date

Oct 2015

Definition at line 1179 of file mo_julian.f90.

Referenced by caldat(), and dec2date360().

Here is the caller graph for this function:

15.83.2.3 elemental subroutine mo_julian::caldat365 (integer(i4), intent(in) *julian,* integer(i4), intent(out) *dd,* integer(i4), intent(out) *mm,* integer(i4), intent(out) *yy*) [private]

Day, month and year from Julian day in a 365 day calendar.

Inverse of the function julday365. Here julian is input as a Julian Day Number, and the routine outputs dd, mm, and yy as the day, month, and year on which the specified Julian Day started at noon. The zeroth Julian Day here is 01.01.0000

Parameters

in	*integer(i4) :: julday*	Julian day
out	*integer(i4) :: dd*	Day in month of Julian day
out	*integer(i4) :: mm*	Month in year of Julian day
out	*integer(i4) :: yy*	Year of Julian day

Author

Written, David Schaefer

Date

Dec 2015

Definition at line 1470 of file mo_julian.f90.

Referenced by caldat(), and dec2date365().

Here is the caller graph for this function:

15.83.2.4 elemental subroutine, public mo_julian::caldatjulian (integer(i4), intent(in) *julian,* integer(i4), intent(out) *dd,* integer(i4), intent(out) *mm,* integer(i4), intent(out) *yy*)

Day, month and year from Julian day.

Inverse of the function juldayJulian. Here julian is input as a Julian Day Number, and the routine outputs id, mm, and yy as the day, month, and year on which the specified Julian Day started at noon. The zeroth Julian Day is 01.01.-4712, i.e. the 1st January 4713 BC. Julian day definition starts at 1st January 4713 BC.
Here, the astronomical definition is used, i.e. the year 1 BC (historic) is counted as 0 (astronomic), 2 BC is -1, etc.
This means that Julian day definition starts as 01.01.-4712 in astronomical units.

Parameters

in	*integer(i4) :: Julday*	Julian day
out	*integer(i4) :: dd*	Day in month of Julian day
out	*integer(i4) :: mm*	Month in year of Julian day
out	*integer(i4) :: yy*	Year of Julian day

Note

Julian day definition starts at noon of the 1st January 4713 BC.
Here, the astronomical definition is used, i.e. the year 1 BC (historic) is counted as 0 (astronomic), 2 BC is -1, etc.
This means that Julian day definition starts as 01.01.-4712 in astronomical units.

julday and caldat start at midnight of the 1st January 4713 BC. So date2decJulian and juldayJulian as well as dec2dateJulian and caldatJulian are shifted by half a day.
Use date2decJulian with dec2dateJulian together for fractional Julian dates and use juldayJulian with caldat↩ Julian together for integer Julian days.

Author

Written, Matthias Cuntz - modified julday from Numerical Recipes

Date

Dec 2011 Modified Matthias Cuntz, May 2014 - changed to new algorithm with astronomical units removed numerical recipes David Schaefer, Jan 2016 - renamed procodure

Definition at line 555 of file mo_julian.f90.

Referenced by caldat().

Here is the caller graph for this function:

15.83.2.5 elemental real(dp) function, public mo_julian::date2dec (integer(i4), intent(in), optional *dd,* integer(i4), intent(in), optional *mm,* integer(i4), intent(in), optional *yy,* integer(i4), intent(in), optional *hh,* integer(i4), intent(in), optional *nn,* integer(i4), intent(in), optional *ss,* integer(i4), intent(in), optional *calendar*)

Fractional Julian day from day, month, year, hour, minute, second in the current calendar.

Wrapper around the calendar specific date2dec procedures. In this routine date2dec returns the fractional Julian Day that begins at noon of the calendar date specified by month mm, day dd, and year yy, all integer variables. The zeroth Julian Day depends on the called procedure. See their documentation for details.

Parameters

in	*integer(i4), optional :: dd*	Day in month of Julian day (default: 1)
in	*integer(i4), optional :: mm*	Month in year of Julian day (default: 1)
in	*integer(i4), optional :: yy*	Year of Julian day (default: 1)
in	*integer(i4), optional :: hh*	Hours of Julian day (default: 0)
in	*integer(i4), optional :: nn*	Minutes of hour of Julian day (default: 0)
in	*integer(i4), optional :: ss*	Secondes of minute of hour of Julian day (default: 0)

in	*integer(i4), optional :: calendar*	The calendar to use, the global calendar will be used by default

Returns

real(dp) :: date2dec ! Fractional Julian day

Author

Written, David Schaefer

Date

Jan 2015

Definition at line 399 of file mo_julian.f90.

References date2dec360(), date2dec365(), date2decjulian(), and selectcalendar().

Referenced by mo_mrm_signatures::maximummonthlyflow(), and mo_mhm::mhm().

Here is the call graph for this function:

Here is the caller graph for this function:

15.83.2.6 elemental real(dp) function mo_julian::date2dec360 (integer(i4), intent(in), optional *dd,* integer(i4), intent(in), optional *mm,* integer(i4), intent(in), optional *yy,* integer(i4), intent(in), optional *hh,* integer(i4), intent(in), optional *nn,* integer(i4), intent(in), optional *ss*) [private]

Fractional Julian day from day, month, year, hour, minute, second in 360 day calendar.

In this routine date2dec360 returns the fractional Julian Day that begins at noon of the calendar date specified by month mm, day dd, and year yy, all integer variables. The zeroth Julian Day is 01.01.0000 at noon.

Parameters

in	*integer(i4), optional :: dd*	Day in month of Julian day (default: 1)
in	*integer(i4), optional :: mm*	Month in year of Julian day (default: 1)
in	*integer(i4), optional :: yy*	Year of Julian day (default: 1)
in	*integer(i4), optional :: hh*	Hours of Julian day (default: 0)
in	*integer(i4), optional :: nn*	Minutes of hour of Julian day (default: 0)
in	*integer(i4), optional :: ss*	Secondes of minute of hour of Julian day (default: 0)

Returns

real(dp) :: date2dec360 ! Fractional Julian day

Author

Written, David Schaefer

Date

Oct 2015

Definition at line 1390 of file mo_julian.f90.

Referenced by date2dec().

Here is the caller graph for this function:

15.83.2.7 elemental real(dp) function mo_julian::date2dec365 (integer(i4), intent(in), optional *dd,* integer(i4), intent(in), optional *mm,* integer(i4), intent(in), optional *yy,* integer(i4), intent(in), optional *hh,* integer(i4), intent(in), optional *nn,* integer(i4), intent(in), optional *ss*) [private]

Fractional Julian day from day, month, year, hour, minute, second in 365 day calendar.

In this routine date2dec365 returns the fractional Julian Day that begins at noon of the calendar date specified by month mm, day dd, and year yy, all integer variables. The zeroth Julian Day is 01.01.0000 at noon.

Parameters

in	*integer(i4), optional :: dd*	Day in month of Julian day (default: 1)
in	*integer(i4), optional :: mm*	Month in year of Julian day (default: 1)
in	*integer(i4), optional :: yy*	Year of Julian day (default: 1)
in	*integer(i4), optional :: hh*	Hours of Julian day (default: 0)
in	*integer(i4), optional :: nn*	Minutes of hour of Julian day (default: 0)
in	*integer(i4), optional :: ss*	Secondes of minute of hour of Julian day (default: 0)

Returns

real(dp) :: date2dec365 ! Fractional Julian day

Author

Written, David Schaefer

Date

Dec 2015

Definition at line 1691 of file mo_julian.f90.

Referenced by date2dec().

Here is the caller graph for this function:

15.83.2.8 elemental real(dp) function mo_julian::date2decjulian (integer(i4), intent(in), optional *dd,* integer(i4), intent(in), optional *mm,* integer(i4), intent(in), optional *yy,* integer(i4), intent(in), optional *hh,* integer(i4), intent(in), optional *nn,* integer(i4), intent(in), optional *ss*) [private]

Fractional Julian day from day, month, year, hour, minute, second.

In this routine date2decJulian returns the fractional Julian Day that begins at noon of the calendar date specified by month mm, day dd, and year yy, all integer variables. The zeroth Julian Day is 01.01.-4712 at noon, i.e. the 1st January 4713 BC 12:00:00 h. Julian day definition starts at noon of the 1st January 4713 BC.
Here, the astronomical definition is used, i.e. the year 1 BC (historic) is counted as 0 (astronomic), 2 BC is -1, etc. This means that Julian day definition starts as 01.01.-4712 in astronomical units.

Parameters

in	*integer(i4), optional :: dd*	Day in month of Julian day (default: 1)
in	*integer(i4), optional :: mm*	Month in year of Julian day (default: 1)
in	*integer(i4), optional :: yy*	Year of Julian day (default: 1)
in	*integer(i4), optional :: hh*	Hours of Julian day (default: 0)
in	*integer(i4), optional :: nn*	Minutes of hour of Julian day (default: 0)
in	*integer(i4), optional :: ss*	Secondes of minute of hour of Julian day (default: 0)

Returns

real(dp) :: date2dec — Fractional Julian day

Note

Julian day definition starts at noon of the 1st January 4713 BC.
Here, the astronomical definition is used, i.e. the year 1 BC (historic) is counted as 0 (astronomic), 2 BC is -1, etc.
This means that Julian day definition starts as 01.01.-4712 in astronomical units.

juldayJulian and caldatJulian start at midnight of the 1st January 4713 BC. So date2decJulian and juldayJulian as well as dec2dateJulian and caldatJulian are shifted by half a day.
Use date2decJulian with dec2dateJulian together for fractional Julian dates and use juldayJulian with caldat↩ Julian together for integer Julian days.

Author

Written, Matthias Cuntz

Date

Jan 2013 Modified Matthias Cuntz, May 2014 - changed to new algorithm with astronomical units removed numerical recipes David Schaefer, Jan 2016 - renamed procodure

Definition at line 673 of file mo_julian.f90.

Referenced by date2dec().

Here is the caller graph for this function:

15.83.2.9 elemental subroutine, public mo_julian::dec2date (real(dp), intent(in) *julian,* integer(i4), intent(out), optional *dd,* integer(i4), intent(out), optional *mm,* integer(i4), intent(out), optional *yy,* integer(i4), intent(out), optional *hh,* integer(i4), intent(out), optional *nn,* integer(i4), intent(out), optional *ss,* integer(i4), intent(in), optional *calendar*)

Day, month, year, hour, minute, and second from fractional Julian day in the current or given calendar.

Wrapper around the calendar specific dec2date procedures. Inverse of the function date2dec. Here dec2date is input as a fractional Julian Day. The routine outputs dd, mm, yy, hh, nn, ss as the day, month, year, hour, minute, and second on which the specified Julian Day started at noon. The zeroth Julian Day depends on the called procedure. See their documentation for details.

Parameters

in	*real(dp) :: fJulian*	fractional Julian day
in	*integer(i4) :: calendar*	The calendar to use, the global calendar will be used by default
out	*integer(i4), optional :: dd*	Day in month of Julian day
out	*integer(i4), optional :: mm*	Month in year of Julian day
out	*integer(i4), optional :: yy*	Year of Julian day
out	*integer(i4), optional :: hh*	Hour of Julian day
out	*integer(i4), optional :: nn*	Minute in hour of Julian day
out	*integer(i4), optional :: ss*	Second in minute of hour of Julian day

Author

Written, David Schaefer

Date

Jan 2015

Definition at line 329 of file mo_julian.f90.

References dec2date360(), dec2date365(), dec2datejulian(), and selectcalendar().

Referenced by mo_mrm_river_head::create_output(), mo_write_fluxes_states::createoutputfile(), mo_mrm_write_fluxes_states::createoutputfile(), mo_temporal_aggregation::day2mon_average_dp(), mo_read_forcing_nc::get_time_vector_and_select(), mo_temporal_aggregation::hour2day_average_dp(), mo_mrm_signatures::maximummonthlyflow(), mo_mhm::mhm(), mo_mrm_objective_function_runoff::multi_objective_ae_fdc_lsv_nse_djf(), and mo_mrm_write::write_daily_obs_sim_discharge().

Here is the call graph for this function:

Here is the caller graph for this function:

15.83.2.10 elemental subroutine mo_julian::dec2date360 (real(dp), intent(in) *julian,* integer(i4), intent(out), optional *dd,* integer(i4), intent(out), optional *mm,* integer(i4), intent(out), optional *yy,* integer(i4), intent(out), optional *hh,* integer(i4), intent(out), optional *nn,* integer(i4), intent(out), optional *ss*) [private]

Day, month, year, hour, minute, and second from fractional Julian day in a 360_day calendar.

Inverse of the function date2dec360. Here dec2date360 is input as a fractional Julian Day. The routine outputs dd, mm, yy, hh, nn, ss as the day, month, year, hour, minute, and second on which the specified Julian Day started at noon. The zeroth Julian Day is 01.01.0000 at noon.

Parameters

in	*real(dp) :: fJulian*	fractional Julian day
out	*integer(i4), optional :: dd*	Day in month of Julian day
out	*integer(i4), optional :: mm*	Month in year of Julian day
out	*integer(i4), optional :: yy*	Year of Julian day
out	*integer(i4), optional :: hh*	Hour of Julian day
out	*integer(i4), optional :: nn*	Minute in hour of Julian day
out	*integer(i4), optional :: ss*	Second in minute of hour of Julian day

Author

Written, David Schaefer

Date

Oct 2015

Definition at line 1299 of file mo_julian.f90.

References caldat360(), and julday360().

Referenced by dec2date().

Here is the call graph for this function:

Here is the caller graph for this function:

15.83.2.11 elemental subroutine mo_julian::dec2date365 (real(dp), intent(in) *julian,* integer(i4), intent(out), optional *dd,* integer(i4), intent(out), optional *mm,* integer(i4), intent(out), optional *yy,* integer(i4), intent(out), optional *hh,* integer(i4), intent(out), optional *nn,* integer(i4), intent(out), optional *ss*) [private]

Day, month, year, hour, minute, and second from fractional Julian day in a 365_day calendar.

Inverse of the function date2dec. Here dec2date365 is input as a fractional Julian Day. The routine outputs dd, mm, yy, hh, nn, ss as the day, month, year, hour, minute, and second on which the specified Julian Day started at noon. The zeroth Julian Day is 01.01.0000 at noon.

Parameters

in	*real(dp) :: fJulian*	fractional Julian day
out	*integer(i4), optional :: dd*	Day in month of Julian day
out	*integer(i4), optional :: mm*	Month in year of Julian day

out	*integer(i4), optional :: yy*	Year of Julian day
out	*integer(i4), optional :: hh*	Hour of Julian day
out	*integer(i4), optional :: nn*	Minute in hour of Julian day
out	*integer(i4), optional :: ss*	Second in minute of hour of Julian day

Author

Written, David Schaefer

Date

Dec 2015

Definition at line 1600 of file mo_julian.f90.

References caldat365(), and julday365().

Referenced by dec2date().

Here is the call graph for this function:

Here is the caller graph for this function:

15.83.2.12 elemental subroutine mo_julian::dec2datejulian (real(dp), intent(in) *julian,* integer(i4), intent(out), optional *dd,* integer(i4), intent(out), optional *mm,* integer(i4), intent(out), optional *yy,* integer(i4), intent(out), optional *hh,* integer(i4), intent(out), optional *nn,* integer(i4), intent(out), optional *ss*) [private]

Day, month, year, hour, minute, and second from fractional Julian day.

Inverse of the function date2decJulian. Here dec2dateJulian is input as a fractional Julian Day, which starts at noon of the 1st January 4713 BC, i.e. 01.01.-4712. The routine outputs dd, mm, yy, hh, nn, ss as the day, month, year, hour, minute, and second on which the specified Julian Day started at noon. The zeroth Julian Day is 01.01.-4712 at noon, i.e. the 1st January 4713 BC at noon. Julian day definition starts at 1st January 4713 BC.
Here, the astronomical definition is used, i.e. the year 1 BC (historic) is counted as 0 (astronomic), 2 BC is -1, etc.
This means that Julian day definition starts as 01.01.-4712 in astronomical units.

Parameters

in	*real(dp) :: fJulian*	fractional Julian day
out	*integer(i4), optional :: dd*	Day in month of Julian day
out	*integer(i4), optional :: mm*	Month in year of Julian day
out	*integer(i4), optional :: yy*	Year of Julian day
out	*integer(i4), optional :: hh*	Hour of Julian day
out	*integer(i4), optional :: nn*	Minute in hour of Julian day
out	*integer(i4), optional :: ss*	Second in minute of hour of Julian day

Note

Julian day definition starts at noon of the 1st January 4713 BC.
Here, the astronomical definition is used, i.e. the year 1 BC (historic) is counted as 0 (astronomic), 2 BC is -1, etc.
This means that Julian day definition starts as 01.01.-4712 in astronomical units.

juldayJulian and caldatJulian start at midnight of the 1st January 4713 BC. So date2decJulian and juldayJulian as well as dec2dateJulian and caldatJulian are shifted by half a day.
Use date2decJulian with dec2dateJulian together for fractional Julian dates and use juldayJulian with caldat↩ Julian together for integer Julian days.

Author

Written, Matthias Cuntz

Date

Jan 2013 Modified Matthias Cuntz, May 2014 - changed to new algorithm with astronomical units removed numerical recipes David Schaefer, Jan 2016 - renamed procodure

Definition at line 812 of file mo_julian.f90.

References caldat(), and julday().

Referenced by dec2date().

Here is the call graph for this function:

Here is the caller graph for this function:

15.83.2.13 elemental integer(i4) function, public mo_julian::julday (integer(i4), intent(in) *dd,* integer(i4), intent(in) *mm,* integer(i4), intent(in) *yy,* integer(i4), intent(in), optional *calendar*)

Julian day from day, month and year in the current or given calendar.

Wrapper around the calendar specific julday procedures. In this routine julday returns the Julian Day Number that begins at noon of the calendar date specified by month mm, day dd, and year yy, all integer variables. The zeroth Julian Day depends on the called procedure. See their documentation for details.

Parameters

in	*integer(i4) :: dd*	Day in month of Julian day
in	*integer(i4) :: mm*	Month in year of Julian day
in	*integer(i4) :: yy*	Year of Julian day
in	*integer(i4), optional :: calendar*	The calendar to use, the global calendar will be used by default

Returns

integer(i4) :: julian ! Julian day

Author

Written, David Schaefer

Date

Jan 2015

Definition at line 466 of file mo_julian.f90.

References julday360(), julday365(), juldayjulian(), and selectcalendar().

Referenced by mo_meteo_forcings::chunk_size(), mo_temporal_aggregation::day2mon_average_dp(), dec2datejulian(), mo_read_forcing_nc::get_time_vector_and_select(), mo_temporal_aggregation::hour2day_average_dp(), mo_mhm_eval::mhm_eval(), mo_mrm_eval::mrm_eval(), ndays(), and mo_read_timeseries::read_timeseries().

Here is the call graph for this function:

Here is the caller graph for this function:

15.83.2.14 elemental integer(i4) function mo_julian::julday360 (integer(i4), intent(in) *dd,* integer(i4), intent(in) *mm,* integer(i4), intent(in) *yy*) [private]

Julian day from day, month and year in a 360_day calendar.

In this routine julday360 returns the Julian Day Number that begins at noon of the calendar date specified by month mm, day dd, and year yy, all integer variables. The zeroth Julian Day is 01.01.0000

Parameters

in	*integer(i4) :: dd*	Day in month of Julian day
in	*integer(i4) :: mm*	Month in year of Julian day
in	*integer(i4) :: yy*	Year of Julian day

Returns

integer(i4) :: julian ! Julian day

Author

Written, David Schaefer

Date

Oct 2015

Definition at line 1240 of file mo_julian.f90.

Referenced by dec2date360(), and julday().

Here is the caller graph for this function:

15.83.2.15 elemental integer(i4) function mo_julian::julday365 (integer(i4), intent(in) *dd,* integer(i4), intent(in) *mm,* integer(i4), intent(in) *yy*) [private]

Julian day from day, month and year in a 365_day calendar.

In this routine julday365 returns the Julian Day Number that begins at noon of the calendar date specified by month mm, day dd, and year yy, all integer variables. The zeroth Julian Day is 01.01.0000

Parameters

in	*integer(i4) :: dd*	Day in month of Julian day
in	*integer(i4) :: mm*	Month in year of Julian day
in	*integer(i4) :: yy*	Year of Julian day

Returns

integer(i4) :: julian ! Julian day

Author

Written, David Schaefer

Date

Dec 2015

Definition at line 1539 of file mo_julian.f90.

Referenced by dec2date365(), and julday().

Here is the caller graph for this function:

15.83.2.16 elemental integer(i4) function mo_julian::juldayjulian (integer(i4), intent(in) *dd,* integer(i4), intent(in) *mm,* integer(i4), intent(in) *yy*) [private]

Julian day from day, month and year.

In this routine juldayJulian returns the Julian Day Number that begins at noon of the calendar date specified by month mm, day dd, and year yy, all integer variables. The zeroth Julian Day is 01.01.-4712 at noon, i.e. the 1st January 4713 BC 12:00:00 h. Julian day definition starts at noon of the 1st January 4713 BC.
Here, the astronomical definition is used, i.e. the year 1 BC (historic) is counted as 0 (astronomic), 2 BC is -1, etc.
This means that Julian day definition starts as 01.01.-4712 in astronomical units.

Parameters

in	*integer(i4) :: dd*	Day in month of Julian day
in	*integer(i4) :: mm*	Month in year of Julian day
in	*integer(i4) :: yy*	Year of Julian day

Returns

integer(i4) :: julian — Julian day

Note

Julian day definition starts at noon of the 1st January 4713 BC.
Here, the astronomical definition is used, i.e. the year 1 BC (historic) is counted as 0 (astronomic), 2 BC is -1, etc.
This means that Julian day definition starts as 01.01.-4712 in astronomical units.

juldayJulian and caldatJulian start at midnight of the 1st January 4713 BC. So date2decJulian and juldayJulian as well as dec2dateJulian and caldatJulian are shifted by half a day.
Use date2decJulian with dec2dateJulian together for fractional Julian dates and use juldayJulian with caldat↩ Julian together for integer Julian days.

Author

Written, Matthias Cuntz - modified julday from Numerical Recipes

Date

Dec 2011 Modified Matthias Cuntz, May 2014 - changed to new algorithm with astronomical units removed numerical recipes David Schaefer, Jan 2016 - renamed procodure

Definition at line 967 of file mo_julian.f90.

Referenced by julday().

Here is the caller graph for this function:

15.83.2.17 elemental integer(i4) function, public mo_julian::ndays (integer(i4), intent(in) *dd,* integer(i4), intent(in) *mm,* integer(i4), intent(in) *yy*)

IMSL Julian day from day, month and year.

In this routine ndays returns the IMSL Julian Day Number. Julian days begin at noon of the calendar date specified by month mm, day dd, and year yy, all integer variables. IMSL treats 01.01.1900 as a reference and assigns a Julian day 0 to it. ndays = julday(dd,mm,yy) - julday(01,01,1900)

Parameters

in	*integer(i4) :: dd*	Day in month of IMSL Julian day
in	*integer(i4) :: mm*	Month in year of IMSL Julian day
in	*integer(i4) :: yy*	Year of IMSL Julian day

Returns

integer(i4) :: julian — IMSL Julian day, i.e. days before or after 01.01.1900

Author

Written, Matthias Cuntz

Date

Dec 2011

Definition at line 1059 of file mo_julian.f90.

References julday().

Referenced by mo_meteo_forcings::chunk_size(), mo_temporal_aggregation::day2mon_average_dp(), and mo_↩meteo_forcings::is_read().

Here is the call graph for this function:

Here is the caller graph for this function:

15.83.2.18 elemental subroutine, public mo_julian::ndyin (integer(i4), intent(in) *julian,* integer(i4), intent(out) *dd,* integer(i4), intent(out) *mm,* integer(i4), intent(out) *yy*)

Day, month and year from IMSL Julian day.

Inverse of the function ndys. Here ISML Julian is input as a Julian Day Number minus the Julian Day Number of 01.01.1900, and the routine outputs id, mm, and yy as the day, month, and year on which the specified Julian Day started at noon. ndyin is caldat(IMSLJulian + 2415021, dd, mm, yy)

Parameters

in	*integer(i4) :: julian*	IMSL Julian day, i.e. days before or after 01.01.1900
out	*integer(i4) :: dd*	Day in month of IMSL Julian day
out	*integer(i4) :: mm*	Month in year of IMSL Julian day
out	*integer(i4) :: yy*	Year of IMSL Julian day

Author

Written, Matthias Cuntz

Date

Dec 2011

Definition at line 1123 of file mo_julian.f90.

References caldat().

Here is the call graph for this function:

15.83.2.19 pure integer(i4) function mo_julian::selectcalendar (integer(i4), intent(in), optional *selector*) [private]

Select a calendar.

Returns a valid calendar index, based on the given optional argument and/or the module global private variable calendar. If an invalid selector is passed, its value is ignored and the global calendar value retuned instead.

Parameters

in	*integer(i4), optional :: selector*	Calendar selector {1|2|3}

Author

Written, David Schaefer

Date

Jan 2015

Definition at line 204 of file mo_julian.f90.

Referenced by caldat(), date2dec(), dec2date(), and julday().

Here is the caller graph for this function:

15.83.2.20 subroutine mo_julian::setcalendarinteger (integer(i4), intent(in) *selector*) [private]

Set module private variable calendar.

Parameters

in	*integer(i4) :: selector*	{1|2|3}

Author

Written, David Schaefer

Date

Jan 2015

Definition at line 157 of file mo_julian.f90.

15.83.2.21 subroutine mo_julian::setcalendarstring (character(∗), intent(in) *selector*) [private]

Set module private variable calendar.

Parameters

in	*character(len=∗) :: selector*	{"julian"|"365day"|"360day"}

Author

Written, David Schaefer

Date

Jan 2015

Definition at line 109 of file mo_julian.f90.

References mo_julian::setcalendar::setcalendarinteger().

Here is the call graph for this function:

15.83.3 Member Data Documentation

15.83.3.1 integer(i4), save, private mo_julian::calendar = 1 [private]

Definition at line 68 of file mo_julian.f90.

The documentation for this module was generated from the following file:

- mo_julian.f90

15.84 mo_kind Module Reference

Define number representations.

Collaboration diagram for mo_kind:

Data Types

- type sprs2_dp

  Double Precision Numerical Recipes types for sparse arrays.
- type sprs2_sp

  Single Precision Numerical Recipes types for sparse arrays.

Public Attributes

- integer, parameter i1 = SELECTED_INT_KIND(2)

  1 Byte Integer Kind
- integer, parameter i2 = c_short

  2 Byte Integer Kind
- integer, parameter i4 = c_int

  4 Byte Integer Kind
- integer, parameter i8 = c_long_long

  8 Byte Integer Kind
- integer, parameter sp = c_float

  Single Precision Real Kind.
- integer, parameter dp = c_double

  Double Precision Real Kind.
- integer, parameter spc = c_float_complex

  Single Precision Complex Kind.
- integer, parameter dpc = c_double_complex

  Double Precision Complex Kind.
- integer, parameter lgt = KIND(.true.)

  Logical Kind.

15.84.1 Detailed Description

Define number representations.

This module declares the desired ranges and precisions of the number representations, such as single precision or double precision, 32-bit or 46-bit integer, etc. It confirms mostly with the nrtype module of Numerical Recipes in f90.

Authors

Juliane Mai, Matthias Cuntz, Nov 2011

Date

2011-2014

Copyright

GNU Lesser General Public License http://www.gnu.org/licenses/

Definition at line 46 of file mo_kind.f90.

15.84.2 Member Data Documentation

15.84.2.1 integer, parameter mo_kind::dp = c_double

Double Precision Real Kind.

Definition at line 78 of file mo_kind.f90.

15.84.2.2 integer, parameter mo_kind::dpc = c_double_complex

Double Precision Complex Kind.

Definition at line 86 of file mo_kind.f90.

15.84.2.3 integer, parameter mo_kind::i1 = SELECTED_INT_KIND(2)

1 Byte Integer Kind

Definition at line 57 of file mo_kind.f90.

15.84.2.4 integer, parameter mo_kind::i2 = c_short

2 Byte Integer Kind

Definition at line 62 of file mo_kind.f90.

15.84.2.5 integer, parameter mo_kind::i4 = c_int

4 Byte Integer Kind

Definition at line 66 of file mo_kind.f90.

15.84.2.6 integer, parameter mo_kind::i8 = c_long_long

8 Byte Integer Kind

Definition at line 70 of file mo_kind.f90.

15.84.2.7 integer, parameter mo_kind::lgt = KIND(.true.)

Logical Kind.

Definition at line 88 of file mo_kind.f90.

15.84.2.8 integer, parameter mo_kind::sp = c_float

Single Precision Real Kind.

Definition at line 74 of file mo_kind.f90.

15.84.2.9 integer, parameter mo_kind::spc = c_float_complex

Single Precision Complex Kind.

Definition at line 82 of file mo_kind.f90.

The documentation for this module was generated from the following file:

- mo_kind.f90

15.85 mo_linfit Module Reference

Fitting a straight line.

Data Types

- interface linfit

  Fits a straight line to input data by minimizing chi^2.

Private Member Functions

- real(dp) function, dimension(:),
  allocatable linfit_dp (x, y, a, b, siga, sigb, chi2, model2)
- real(sp) function, dimension(:),
  allocatable linfit_sp (x, y, a, b, siga, sigb, chi2, model2)

15.85.1 Detailed Description

Fitting a straight line.

This module provides a routine to fit a straight line with model I or model II regression.

Authors

Matthias Cuntz

Date

Mar 2011

Definition at line 10 of file mo_linfit.f90.

15.85.2 Member Function/Subroutine Documentation

15.85.2.1 real(dp) function, dimension(:), allocatable mo_linfit::linfit_dp (real(dp), dimension(:), intent(in) *x*, real(dp), dimension(:), intent(in) *y*, real(dp), intent(out), optional *a*, real(dp), intent(out), optional *b*, real(dp), intent(out), optional *siga*, real(dp), intent(out), optional *sigb*, real(dp), intent(out), optional *chi2*, logical, intent(in), optional *model2*) [private]

Definition at line 120 of file mo_linfit.f90.

References mo_linfit::linfit::linfit_dp().

Here is the call graph for this function:

15.85.2.2 real(sp) function, dimension(:), allocatable mo_linfit::linfit_sp (real(sp), dimension(:), intent(in) *x*, real(sp), dimension(:), intent(in) *y*, real(sp), intent(out), optional *a*, real(sp), intent(out), optional *b*, real(sp), intent(out), optional *siga*, real(sp), intent(out), optional *sigb*, real(sp), intent(out), optional *chi2*, logical, intent(in), optional *model2*) [private]

Definition at line 211 of file mo_linfit.f90.

References mo_linfit::linfit::linfit_sp().

Here is the call graph for this function:

The documentation for this module was generated from the following file:

- mo_linfit.f90

15.86 mo_mad Module Reference

Data Types

- interface mad

Private Member Functions

- logical function, dimension(size(arr)) mad_dp (arr, z, mask, deriv)
- logical function, dimension(size(arr)) mad_sp (arr, z, mask, deriv)
- real(dp) function, dimension(size(arr)) mad_val_dp (arr, z, mask, tout, mval)
- real(sp) function, dimension(size(arr)) mad_val_sp (arr, z, mask, tout, mval)

15.86.1 Detailed Description

Definition at line 1 of file mo_mad.f90.

15.86.2 Member Function/Subroutine Documentation

15.86.2.1 logical function, dimension(size(arr)) mo_mad::mad_dp (real(dp), dimension(:), intent(in) *arr,* real(dp), intent(in), optional *z,* logical, dimension(:), intent(in), optional *mask,* integer(i4), intent(in), optional *deriv*) [private]

Definition at line 189 of file mo_mad.f90.

References mo_mad::mad::mad_dp().

Here is the call graph for this function:

15.86.2.2 logical function, dimension(size(arr)) mo_mad::mad_sp (real(sp), dimension(:), intent(in) *arr,* real(sp), intent(in), optional *z,* logical, dimension(:), intent(in), optional *mask,* integer(i4), intent(in), optional *deriv*) [private]

Definition at line 257 of file mo_mad.f90.

References mo_mad::mad::mad_sp().

Here is the call graph for this function:

15.86.2.3 real(dp) function, dimension(size(arr)) mo_mad::mad_val_dp (real(dp), dimension(:), intent(in) *arr,* real(dp), intent(in), optional *z,* logical, dimension(:), intent(in), optional *mask,* character(1) *tout,* real(dp), intent(in), optional *mval*) [private]

Definition at line 326 of file mo_mad.f90.

References mo_mad::mad::mad_val_dp().

Here is the call graph for this function:

15.86.2.4 real(sp) function, dimension(size(arr)) mo_mad::mad_val_sp (real(sp), dimension(:), intent(in) *arr,* real(sp), intent(in), optional *z,* logical, dimension(:), intent(in), optional *mask,* character(1) *tout,* real(sp), intent(in), optional *mval*) [private]

Definition at line 389 of file mo_mad.f90.

References mo_mad::mad::mad_val_sp().

Here is the call graph for this function:

The documentation for this module was generated from the following file:

- mo_mad.f90

15.87 mo_mcmc Module Reference

This module is Monte Carlo Markov Chain sampling of a posterior parameter distribution.

Data Types

- interface mcmc

  This module is Monte Carlo Markov Chain sampling of a posterior parameter distribution (measurement errors are either known or modeled).
- interface mcmc_stddev

  This module is Monte Carlo Markov Chain sampling of a posterior parameter distribution (measurement errors are neither known nor modeled).

Public Member Functions

- real(dp) function pargennorm_dp (old, dMax, oMin, oMax, RN, inbound)
- recursive subroutine generatenewparameterset_dp (ParaSelectMode, paraold, truepara, rangePar, stepsize, save_state_2, save_state_3, paranew, ChangePara)

Private Member Functions

- subroutine mcmc_dp (eval, likelihood, para, rangePar,mcmc_paras, burnin_paras,seed_in, printflag_in, maskpara_in,restart, restart_file,tmp_file,
- subroutine mcmc_stddev_dp (eval, likelihood, para, rangePar,mcmc_paras, burnin_paras,seed_in, printflag↩_in, maskpara_in,tmp_file,
- real(dp) function pargen_dp (old, dMax, oMin, oMax, RN, inbound)

15.87.1 Detailed Description

This module is Monte Carlo Markov Chain sampling of a posterior parameter distribution.

This module is Monte Carlo Markov Chain sampling of a posterior parameter distribution.

Authors

Maren Goehler, Juliane Mai

Date

Aug 2012

Definition at line 10 of file mo_mcmc.f90.

15.87.2 Member Function/Subroutine Documentation

15.87.2.1 recursive subroutine mo_mcmc::generatenewparameterset_dp (integer(i4), intent(in) *ParaSelectMode,* real(dp), dimension(:), intent(in) *paraold,* integer(i4), dimension(:), intent(in) *truepara,* real(dp), dimension(:, :), intent(in) *rangePar,* real(dp), dimension(:), intent(in) *stepsize,* integer(i8), dimension(n_save_state), intent(inout) *save_state_2,* integer(i8), dimension(n_save_state), intent(inout) *save_state_3,* real(dp), dimension(size(paraold)), intent(out) *paranew,* logical, dimension(size(paraold)), intent(out) *ChangePara*)

Definition at line 2132 of file mo_mcmc.f90.

References pargennorm_dp().

Referenced by mcmc_dp(), and mcmc_stddev_dp().

Here is the call graph for this function:

Here is the caller graph for this function:

15.87.2.2 **subroutine mo_mcmc::mcmc_dp (procedure(eval_interface), intent(in), pointer *eval,* procedure(objective_interface), intent(in), pointer *likelihood,* real(dp), dimension(:), intent(in) *para,* real(dp), dimension(:, :), intent(in) *rangePar,* real(dp), dimension(:, :), intent(out), allocatable *mcmc_paras,* real(dp), dimension(:, :), intent(out), allocatable *burnin_paras,* integer(i8), intent(in), optional *seed_in,* logical, intent(in), optional *printflag_in,* logical, dimension(size(para, 1)), intent(in), optional *maskpara_in,* logical, intent(in), optional *restart,* character(len = ∗), intent(in), optional *restart_file,* character(len = ∗), intent(in), optional *tmp_file*)** `[private]`

Definition at line 483 of file mo_mcmc.f90.

References generatenewparameterset_dp().

Here is the call graph for this function:

15.87.2.3 **subroutine mo_mcmc::mcmc_stddev_dp (procedure(eval_interface), intent(in), pointer *eval,* procedure(objective_interface), intent(in), pointer *likelihood,* real(dp), dimension(:), intent(in) *para,* real(dp), dimension(:, :), intent(in) *rangePar,* real(dp), dimension(:, :), intent(out), allocatable *mcmc_paras,* real(dp), dimension(:, :), intent(out), allocatable *burnin_paras,* integer(i8), intent(in), optional *seed_in,* logical, intent(in), optional *printflag_in,* logical, dimension(size(para, 1)), intent(in), optional *maskpara_in,* character(len = ∗), intent(in), optional *tmp_file*)** `[private]`

Definition at line 1338 of file mo_mcmc.f90.

References generatenewparameterset_dp().

Here is the call graph for this function:

15.87.2.4 real(dp) function mo_mcmc::pargen_dp (real(dp), intent(in) *old,* real(dp), intent(in) *dMax,* real(dp), intent(in) *oMin,* real(dp), intent(in) *oMax,* real(dp), intent(in) *RN,* logical, intent(out), optional *inbound*) [private]

Definition at line 2061 of file mo_mcmc.f90.

15.87.2.5 real(dp) function mo_mcmc::pargennorm_dp (real(dp), intent(in) *old,* real(dp), intent(in) *dMax,* real(dp), intent(in) *oMin,* real(dp), intent(in) *oMax,* real(dp), intent(in) *RN,* logical, intent(out), optional *inbound*)

Definition at line 2091 of file mo_mcmc.f90.

Referenced by generatenewparameterset_dp().

Here is the caller graph for this function:

The documentation for this module was generated from the following file:

- mo_mcmc.f90

15.88 mo_message Module Reference

Write out concatenated strings.

Collaboration diagram for mo_message:

Public Member Functions

- subroutine, public message (t01, t02, t03, t04, t05, t06, t07, t08, t09, t10, uni, advance)

  Write out several string concatenated either on screen or in a file.

Public Attributes

- character(len=1024), public message_text = ''

15.88.1 Detailed Description

Write out concatenated strings.

Write out several strings concatenated on standard out or a given unit, either advancing or not.

Author

Matthias Cuntz

Date

Jul 2011

Definition at line 10 of file mo_message.f90.

15.88.2 Member Function/Subroutine Documentation

15.88.2.1 subroutine, public mo_message::message (character(len = ∗), intent(in), optional *t01,* character(len = ∗), intent(in), optional *t02,* character(len = ∗), intent(in), optional *t03,* character(len = ∗), intent(in), optional *t04,* character(len = ∗), intent(in), optional *t05,* character(len = ∗), intent(in), optional *t06,* character(len = ∗), intent(in), optional *t07,* character(len = ∗), intent(in), optional *t08,* character(len = ∗), intent(in), optional *t09,* character(len = ∗), intent(in), optional *t10,* integer, intent(in), optional *uni,* character(len = ∗), intent(in), optional *advance*)

Write out several string concatenated either on screen or in a file.

Parameters

in	*character(len=∗), optional :: t01*	1st string
in	*character(len=∗), optional :: t02*	2nd string
in	*character(len=∗), optional :: t03*	3rd string
in	*character(len=∗), optional :: t04*	4th string
in	*character(len=∗), optional :: t05*	5th string
in	*character(len=∗), optional :: t06*	6th string
in	*character(len=∗), optional :: t07*	7th string
in	*character(len=∗), optional :: t08*	8th string
in	*character(len=∗), optional :: t09*	9th string
in	*character(len=∗), optional :: t10*	10th string

in	*integer , optional :: unit*	Unit to write to (default: nout)
in	*character(len=∗), optional :: advance*	WRITE advance keyword (default: 'yes') yes: newline will be written after message no: no newline at end of message

Author

Matthias Cuntz - modified from Echam5, (C) MPI-MET, Hamburg, Germany

Date

Dec 2011

Definition at line 109 of file mo_message.f90.

Referenced by mo_grid::calculate_grid_properties(), mo_meteo_forcings::chunk_size(), mo_write_fluxes_states↩::close(), mo_mrm_write_fluxes_states::close(), mo_common_read_config::common_read_config(), mo_mrm_↩init::config_output(), mo_startup::constants_init(), mo_mrm_river_head::create_output(), mo_objective_function↩::extract_basin_avg_tws(), mo_mrm_objective_function_runoff::extract_runoff(), mo_read_forcing_nc::get_time↩_vector_and_select(), mo_meteo_forcings::is_read(), mo_mrm_init::l0_check_input_routing(), mo_mrm_net_↩startup::l11_flow_direction(), mo_mrm_net_startup::l11_link_location(), mo_startup::l2_variable_init(), mo_mrm_↩signatures::limb_densities(), mo_mrm_signatures::maximummonthlyflow(), mo_meteo_forcings::meteo_weights↩_wrapper(), mo_mhm::mhm(), mhm_driver(), mo_mhm_eval::mhm_eval(), mo_mhm_read_config::mhm_read↩_config(), mo_mrm_signatures::moments(), mrm_driver(), mo_mrm_eval::mrm_eval(), mo_mrm_init::mrm_init(), mo_mrm_mpr::mrm_init_param(), mo_mrm_read_config::mrm_read_config(), mo_mrm_read_data::mrm_read↩_discharge(), mo_mrm_read_data::mrm_read_l0_data(), mo_mrm_restart::mrm_read_restart_config(), mo_↩mrm_mpr::mrm_update_param(), mo_mrm_write::mrm_write_optifile(), mo_mrm_write::mrm_write_optinamelist(), mo_mrm_restart::mrm_write_restart(), mo_objective_function::objective(), mo_objective_function::objective_et↩_kge_catchment_avg(), mo_objective_function::objective_kge_q_et(), mo_objective_function::objective_kge_q_↩rmse_et(), mo_objective_function::objective_kge_q_rmse_tws(), mo_objective_function::objective_kge_q_sm_↩corr(), mo_objective_function::objective_neutrons_kge_catchment_avg(), mo_objective_function::objective_sm↩_corr(), mo_objective_function::objective_sm_kge_catchment_avg(), mo_objective_function::objective_sm_pd(), mo_objective_function::objective_sm_sse_standard_score(), mo_nml::open_nml(), mo_meteo_forcings::prepare↩_meteo_forcings_data(), mo_mrm_init::print_startup_message(), mo_read_optional_data::read_basin_avg_tws(), mo_read_forcing_nc::read_const_forcing_nc(), mo_common_read_data::read_dem(), mo_read_optional_data↩::read_evapotranspiration(), mo_read_forcing_nc::read_forcing_nc(), mo_common_restart::read_grid_info(), mo↩_read_latlon::read_latlon(), mo_common_read_data::read_lcover(), mo_mrm_read_config::read_mrm_routing_↩params(), mo_read_optional_data::read_neutrons(), mo_read_optional_data::read_soil_moisture(), mo_read_↩timeseries::read_timeseries(), mo_read_forcing_nc::read_weights_nc(), mo_mrm_signatures::runoffratio(), mo↩_common_read_config::set_land_cover_scenes_id(), mo_mrm_objective_function_runoff::single_objective_↩runoff(), mo_mrm_write::write_configfile(), mo_write_ascii::write_configfile(), mo_mrm_write::write_daily_obs_↩sim_discharge(), mo_write_ascii::write_optifile(), mo_write_ascii::write_optinamelist(), mo_restart::write_restart_↩files(), and mo_mrm_signatures::zeroflowratio().

15.88.3 Member Data Documentation

15.88.3.1 character(len = 1024), public mo_message::message_text = ''

Definition at line 45 of file mo_message.f90.

The documentation for this module was generated from the following file:

- mo_message.f90

15.89 mo_meteo_forcings Module Reference

Prepare meteorological forcings data for mHM.

Public Member Functions

- subroutine, public prepare_meteo_forcings_data (iBasin, tt)

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

  Prepare meteorological forcings data for mHM at Level-1.
- subroutine meteo_weights_wrapper (iBasin, read_meteo_weights, dataPath, dataOut1, lower, upper, ncvar↩Name)

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

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

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

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

15.89.1 Detailed Description

Prepare meteorological forcings data for mHM.

Prepare meteorological forcings data for mHM.

Authors

Rohini Kumar

Date

Jan 2012

Definition at line 13 of file mo_meteo_forcings.f90.

15.89.2 Member Function/Subroutine Documentation

15.89.2.1 subroutine mo_meteo_forcings::chunk_config (integer(i4), intent(in) *iBasin,* integer(i4), intent(in) *tt,* logical, intent(out) *read_flag,* type(period), intent(out) *readPer*)

determines the start date, end date, and read_flag given basin id and current timestep

TODO: add description

Parameters

in	*integer(i4) :: iBasin*	current Basin
in	*integer(i4) :: tt*	current timestep
out	*logical :: read_flag*	indicate whether reading data should be read
out	*type(period) :: readPer*	start and end dates of reading Period

Authors

Stephan Thober

Date

Jun 2014

Definition at line 556 of file mo_meteo_forcings.f90.

References chunk_size(), and is_read().

Referenced by prepare_meteo_forcings_data().

Here is the call graph for this function:

Here is the caller graph for this function:

15.89.2.2 subroutine mo_meteo_forcings::chunk_size (integer(i4), intent(in) *iBasin,* integer(i4), intent(in) *tt,* type(period), intent(out) *readPer*)

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

TODO: add description

Parameters

in	*integer(i4) :: iBasin*	current Basin to process
in	*integer(i4) :: tt*	current time step
out	*type(period) :: readPer*	start and end dates of read Period

Authors

Stephan Thober

Date

Jun 2014

Definition at line 731 of file mo_meteo_forcings.f90.

References mo_julian::caldat(), mo_julian::julday(), mo_message::message(), and mo_julian::ndays().

Referenced by chunk_config().

Here is the call graph for this function:

Here is the caller graph for this function:

15.89.2.3 logical function mo_meteo_forcings::is_read (integer(i4), intent(in) *iBasin,* integer(i4), intent(in) *tt*)

evaluate whether new chunk should be read at this timestep

TODO: add description

Parameters

in	*integer(i4) :: iBasin*	current Basin
in	*integer(i4) :: tt*	current time step

Authors

Stephan Thober

Date

Jun 2014

Definition at line 620 of file mo_meteo_forcings.f90.

References mo_julian::caldat(), mo_message::message(), and mo_julian::ndays().

Referenced by chunk_config().

Here is the call graph for this function:

Here is the caller graph for this function:

15.89.2.4 subroutine mo_meteo_forcings::meteo_forcings_wrapper (integer(i4), intent(in) *iBasin,* character(len = ∗), intent(in) *dataPath,* character(len = ∗), intent(in) *inputFormat,* real(dp), dimension(:, :), intent(inout), allocatable *dataOut1,* real(dp), intent(in), optional *lower,* real(dp), intent(in), optional *upper,* character(len = ∗), intent(in), optional *ncvarName*)

Prepare meteorological forcings data for mHM at Level-1.

Prepare meteorological forcings data for mHM, which include 1) Reading meteo. datasets at their native resolution for every basin 2) Perform aggregation or disaggregation of meteo. datasets from their native resolution (level-2) to the required hydrologic resolution (level-1) 3) Pad the above datasets of every basin to their respective global ones

Parameters

in	*integer(i4) :: iBasin*	Basin Id
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 239 of file mo_meteo_forcings.f90.

References mo_read_forcing_nc::read_forcing_nc().

Referenced by prepare_meteo_forcings_data().

Here is the call graph for this function:

Here is the caller graph for this function:

15.89.2.5 subroutine mo_meteo_forcings::meteo_weights_wrapper (integer(i4), intent(in) *iBasin,* logical, intent(in) *read_meteo_weights,* character(len = ∗), intent(in) *dataPath,* real(dp), dimension(:, :, :), intent(inout), allocatable *dataOut1,* real(dp), intent(in), optional *lower,* real(dp), intent(in), optional *upper,* character(len = ∗), intent(in), optional *ncvarName*)

Prepare weights for meteorological forcings data for mHM at Level-1.

Prepare meteorological weights data for mHM, which include 1) Reading meteo. weights datasets at their native resolution for every basin 2) Perform aggregation or disaggregation of meteo. weights datasets from their native resolution (level-2) to the required hydrologic resolution (level-1) 3) Pad the above datasets of every basin to their respective global ones

Parameters

in	*integer(i4) :: iBasin*	Basin Id
in	*logical :: read_meteo_weights*	Flag for reading meteo weights
in	*character(len = *) :: dataPath*	Data path where a given meteo. variable is stored
in,out	*real(dp), dimension(:, :, :) :: dataOut1*	Packed meterological variable for the whole simulation period
in	*real(dp), optional :: lower*	Lower bound for check of validity of data values
in	*real(dp), optional :: upper*	Upper bound for check of validity of data values
in	*character(len = *), optional :: ncvarName*	name of the variable (for .nc files)

Authors

Stephan Thober & Rohini Kumar

Date

Jan 2017

Definition at line 404 of file mo_meteo_forcings.f90.

References mo_message::message(), and mo_read_forcing_nc::read_weights_nc().

Referenced by prepare_meteo_forcings_data().

Here is the call graph for this function:

Here is the caller graph for this function:

15.89.2.6 subroutine, public mo_meteo_forcings::prepare_meteo_forcings_data (integer(i4), intent(in) *iBasin,* integer(i4), intent(in) *tt*)

Prepare meteorological forcings data for a given variable.

Prepare meteorological forcings data for a given variable. Internally this subroutine calls another routine meteo_↩ wrapper for different meterological variables

Parameters

in	*integer(i4) :: iBasin*	Basin Id
in	*integer(i4) :: tt*	current timestep

Authors

Rohini Kumar

Date

Jan 2013

Definition at line 63 of file mo_meteo_forcings.f90.

References chunk_config(), mo_message::message(), meteo_forcings_wrapper(), meteo_weights_wrapper(), mo↩ _timer::timer_get(), mo_timer::timer_start(), and mo_timer::timer_stop().

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:

The documentation for this module was generated from the following file:

- mo_meteo_forcings.f90

15.90 mo_mhm Module Reference

Call all main processes of mHM.

Public Member Functions

- subroutine, public mhm (read_states, tt, time, processMatrix, horizon_depth, nCells1, nHorizons_mH↩M, ntimesteps_day, 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, f↩Asp, 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.

15.90.1 Detailed Description

Call all main processes of mHM.

This module calls all processes of mHM for a given configuration. The configuration of the model is stored in the a process matrix. This configuration is specified in the namelist mhm.nml. The processes are executed in ascending order. At the moment only

process 5 and 8 have options. Currently the following processes are implemented: Process	Name	Flag	Description
1	interception	1	Maximum interception
2	snow and melting	1	Degree-day
3	soil moisture	1	Feddes equation for ET reduction, Brooks-Corey like
3	soil moisture	2	Jarvis equation for ET reduction, Brooks-Corey like
3	soil moisture	3	Jarvis eq. for ET red. + FC dependency on root frac. coef.
4	direct runoff	1	Linear reservoir exceedance
5	PET	-1	PET is input, LAI based correction, dynamic scaling func.
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

Definition at line 39 of file mo_mhm.f90.

15.90.2 Member Function/Subroutine Documentation

15.90.2.1 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), dimension(:), intent(in) *neutron_integral_AFast,* real(dp), dimension(:), intent(in) *global_parameters,* real(dp), dimension(:), intent(in) *latitude,* real(dp), dimension(:), intent(in) *evap_coeff,* real(dp), dimension(:), intent(in) *fday_prec,* real(dp), dimension(:), intent(in) *fnight_prec,* real(dp), dimension(:), intent(in) *fday_pet,* real(dp), dimension(:), intent(in) *fnight_pet,* real(dp), dimension(:), intent(in) *fday_temp,* real(dp), dimension(:), intent(in) *fnight_temp,* real(dp), dimension(:, :, :), intent(in) *temp_weights,* real(dp), dimension(:, :, :), intent(in) *pet_weights,* real(dp), dimension(:, :, :), intent(in) *pre_weights,* logical, intent(in) *read_meteo_weights,* real(dp), dimension(:), intent(in) *pet_in,* real(dp), dimension(:), intent(in) *tmin_in,* real(dp), dimension(:), intent(in) *tmax_in,* real(dp), dimension(:), intent(in) *netrad_in,* real(dp), dimension(:), intent(in) *absvappres_in,* real(dp), dimension(:), intent(in) *windspeed_in,* real(dp), dimension(:), intent(in) *prec_in,* real(dp), dimension(:), intent(in) *temp_in,* real(dp), dimension(:), intent(inout) *fSealed1,* real(dp), dimension(:), intent(inout) *interc,* real(dp), dimension(:), intent(inout) *snowpack,* real(dp), dimension(:), intent(inout) *sealedStorage,* real(dp), dimension(:, :), intent(inout) *soilMoisture,* real(dp), dimension(:), intent(inout) *unsatStorage,* real(dp), dimension(:), intent(inout) *satStorage,* real(dp), dimension(:), intent(inout) *neutrons,* real(dp), dimension(:), intent(inout) *pet_calc,* real(dp), dimension(:, :), intent(inout) *aet_soil,* real(dp), dimension(:), intent(inout) *aet_canopy,* real(dp), dimension(:), intent(inout) *aet_sealed,* real(dp), dimension(:), intent(inout) *baseflow,* real(dp), dimension(:, :), intent(inout) *infiltration,* real(dp), dimension(:), intent(inout) *fast_interflow,* real(dp), dimension(:), intent(inout) *melt,* real(dp), dimension(:), intent(inout) *perc,* real(dp), dimension(:), intent(inout) *prec_effect,* real(dp), dimension(:), intent(inout) *rain,* real(dp), dimension(:), intent(inout) *runoff_sealed,* real(dp), dimension(:), intent(inout) *slow_interflow,* real(dp), dimension(:), intent(inout) *snow,* real(dp), dimension(:), intent(inout) *throughfall,* real(dp), dimension(:), intent(inout) *total_runoff,* real(dp), dimension(:), intent(inout) *alpha,* real(dp), dimension(:), intent(inout) *deg_day_incr,* real(dp), dimension(:), intent(inout) *deg_day_max,* real(dp), dimension(:), intent(inout) *deg_day_noprec,* real(dp), dimension(:), intent(inout) *deg_day,* real(dp), dimension(:), intent(inout) *fAsp,* real(dp), dimension(:), intent(inout) *petLAIcorFactorL1,* real(dp), dimension(:), intent(inout) *HarSamCoeff,* real(dp), dimension(:), intent(inout) *PrieTayAlpha,* real(dp), dimension(:), intent(inout) *aeroResist,* real(dp), dimension(:), intent(inout) *surfResist,* real(dp), dimension(:, :), intent(inout) *frac_roots,* real(dp), dimension(:), intent(inout) *interc_max,* real(dp), dimension(:), intent(inout) *karst_loss,* real(dp), dimension(:), intent(inout) *k0,* real(dp), dimension(:), intent(inout) *k1,* real(dp), dimension(:), intent(inout) *k2,* real(dp), dimension(:), intent(inout) *kp,* real(dp), dimension(:, :), intent(inout) *soil_moist_FC,* real(dp), dimension(:, :), intent(inout) *soil_moist_sat,* real(dp), dimension(:, :), intent(inout) *soil_moist_exponen,* real(dp), dimension(:), intent(inout) *jarvis_thresh_c1,* real(dp), dimension(:), intent(inout) *temp_thresh,* real(dp), dimension(:), intent(inout) *unsat_thresh,* real(dp), dimension(:), intent(inout) *water_thresh_sealed,* real(dp), dimension(:, :), intent(inout) *wilting_point*)

Pure mHM calculations.

Pure mHM calculations. All variables are allocated and initialized. They will be local variables within this call.

Parameters

in	*logical :: read_states*	indicated whether states have been read from file
in	*integer(i4) :: tt*	simulation time step
in	*real(dp) :: time*	current decimal Julian day
in	*integer(i4), dimension(:, :) :: processMatrix*	mHM process configuration matrix
in	*real(dp), dimension(:) :: horizon_depth*	Depth of each horizon in mHM
in	*integer(i4) :: nCells1*	number of cells in a given basin at level L1
in	*integer(i4) :: nHorizons_mHM*	Number of Horizons in mHM
in	*real(dp) :: ntimesteps_day*	number of time intervals per day, transformed in dp
in	*real(dp), dimension(:) :: neutron_↩ integral_AFast*	tabular for neutron flux approximation
in	*real(dp), dimension(:) :: global_↩ parameters*	global mHM parameters
in	*real(dp), dimension(:) :: latitude*	latitude on level 1
in	*real(dp), dimension(:) :: evap_coeff*	Evaporation coefficent for free-water surface of that current month
in	*real(dp), dimension(:) :: fday_prec*	[-] day ratio precipitation $<$ 1
in	*real(dp), dimension(:) :: fnight_prec*	[-] night ratio precipitation $<$ 1
in	*real(dp), dimension(:) :: fday_pet*	[-] day ratio PET $<$ 1
in	*real(dp), dimension(:) :: fnight_pet*	[-] night ratio PET $<$ 1
in	*real(dp), dimension(:) :: fday_temp*	[-] day factor mean temp
in	*real(dp), dimension(:) :: fnight_temp*	[-] night factor mean temp
in	*real(dp), dimension(:, :, :) :: temp_weights*	multiplicative weights for temperature (deg K)

in	*real(dp), dimension(:, :, :) :: pet_weights*	multiplicative weights for potential evapotranspiration
in	*real(dp), dimension(:, :, :) :: pre_weights*	multiplicative weights for precipitation
in	*logical :: read_meteo_weights*	flag whether weights for tavg and pet have read and should be used
in	*real(dp), dimension(:) :: pet_in*	[mm d-1] Daily potential evapotranspiration (input)
in	*real(dp), dimension(:) :: tmin_in*	[degc] Daily minimum temperature
in	*real(dp), dimension(:) :: tmax_in*	[degc] Daily maxumum temperature
in	*real(dp), dimension(:) :: netrad_in*	[w m2] Daily average net radiation
in	*real(dp), dimension(:) :: absvappres_in*	[Pa] Daily average absolute vapour pressure
in	*real(dp), dimension(:) :: windspeed_in*	[m s-1] Daily average wind speed
in	*real(dp), dimension(:) :: prec_in*	[mm d-1] Daily mean precipitation
in	*real(dp), dimension(:) :: temp_in*	[degc] Daily average temperature
in,out	*real(dp), dimension(:) :: fSealed1*	fraction of sealed area at scale L1
in,out	*real(dp), dimension(:) :: interc*	Interception
in,out	*real(dp), dimension(:) :: snowpack*	Snowpack
in,out	*real(dp), dimension(:) :: sealedStorage*	Retention storage of impervious areas
in,out	*real(dp), dimension(:, :) :: soilMoisture*	Soil moisture of each horizon
in,out	*real(dp), dimension(:) :: unsatStorage*	Upper soil storage

in,out	*real(dp), dimension(:) :: satStorage*	Groundwater storage
in,out	*real(dp), dimension(:) :: neutrons*	Ground albedo neutrons
in,out	*real(dp), dimension(:) :: pet_calc*	[mm TST-1] estimated PET (if PET is input = corrected values (fAsp∗PET))
in,out	*real(dp), dimension(:, :) :: aet_soil*	actual ET
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(:) :: petLAIcor↩ FactorL1*	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 199 of file mo_mhm.f90.

References mo_canopy_interc::canopy_interc(), mo_neutrons::cosmic(), mo_julian::date2dec(), mo_julian↩ ::dec2date(), mo_neutrons::desiletsn0(), mo_runoff::l1_total_runoff(), mo_message::message(), mo_pet::pet_↩ hargreaves(), mo_pet::pet_penman(), mo_pet::pet_priestly(), mo_runoff::runoff_sat_zone(), mo_runoff::runoff_↩ unsat_zone(), mo_snow_accum_melt::snow_accum_melt(), mo_soil_moisture::soil_moisture(), and mo_temporal↩ _disagg_forcing::temporal_disagg_forcing().

Referenced by mo_mhm_eval::mhm_eval().

Here is the call graph for this function:

Here is the caller graph for this function:

The documentation for this module was generated from the following file:

- mo_mhm.f90

15.91 mo_mhm_constants Module Reference

Provides mHM specific constants.

Collaboration diagram for mo_mhm_constants:

Public Attributes

- real(dp), parameter, public h2odens = 1000.0_dp
- real(dp), parameter, public p2_initstatefluxes = 15.00_dp
- real(dp), parameter, public p3_initstatefluxes = 10.00_dp
- real(dp), parameter, public p4_initstatefluxes = 75.00_dp
- real(dp), parameter, public p5_initstatefluxes = 1500.00_dp
- real(dp), parameter, public c1_initstatesm = 0.25_dp
- integer(i4), parameter, public noutflxstate = 20_i4
- real(dp), parameter, public stboltzmann = 5.67e-08_dp

  Stefan-Boltzmann constant [W m^-2 K^-4].
- real(dp), parameter, public harsamconst = 17.800_dp

  Hargreaves-Samani ref. ET formula [deg C].
- real(dp), parameter, public duffiedr = 0.0330_dp
- real(dp), parameter, public duffiedelta1 = 0.4090_dp
- real(dp), parameter, public duffiedelta2 = 1.3900_dp
- real(dp), parameter, public tetens_c1 = 0.6108_dp

  Tetens's formula to calculate saturated vapour pressure.
- real(dp), parameter, public tetens_c2 = 17.270_dp
- real(dp), parameter, public tetens_c3 = 237.30_dp
- real(dp), parameter, public satpressureslope1 = 4098.0_dp

  calculation of the slope of the saturation vapour pressure curve following Tetens
- real(dp), parameter, public desilets_a0 = 0.0808_dp

  Neutrons and moisture: N0 formula, Desilets et al. 2010.
- real(dp), parameter, public desilets_a1 = 0.372_dp
- real(dp), parameter, public desilets_a2 = 0.115_dp

- real(dp), parameter, public cosmic_bd = 1.4020_dp

  Neutrons and moisture: COSMIC, Shuttleworth et al. 2013.
- real(dp), parameter, public cosmic_vwclat = 0.0753_dp
- real(dp), parameter, public cosmic_n = 348.33_dp
- real(dp), parameter, public cosmic_alpha = 0.2392421548_dp
- real(dp), parameter, public cosmic_l1 = 161.98621864_dp
- real(dp), parameter, public cosmic_l2 = 129.14558985_dp
- real(dp), parameter, public cosmic_l3 = 107.82204562_dp
- real(dp), parameter, public cosmic_l4 = 3.1627190566_dp

15.91.1 Detailed Description

Provides mHM specific constants.

Provides mHM specific constants such as flood plain elevation.

Authors

Matthias Cuntz

Date

Nov 2011

Definition at line 14 of file mo_mhm_constants.f90.

15.91.2 Member Data Documentation

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

Definition at line 30 of file mo_mhm_constants.f90.

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

Definition at line 62 of file mo_mhm_constants.f90.

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

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

Definition at line 59 of file mo_mhm_constants.f90.

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

Definition at line 63 of file mo_mhm_constants.f90.

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

Definition at line 64 of file mo_mhm_constants.f90.

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

Definition at line 65 of file mo_mhm_constants.f90.

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

Definition at line 66 of file mo_mhm_constants.f90.

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

Definition at line 61 of file mo_mhm_constants.f90.

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

Definition at line 60 of file mo_mhm_constants.f90.

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

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

Definition at line 54 of file mo_mhm_constants.f90.

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

Definition at line 55 of file mo_mhm_constants.f90.

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

Definition at line 56 of file mo_mhm_constants.f90.

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

Definition at line 43 of file mo_mhm_constants.f90.

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

Definition at line 44 of file mo_mhm_constants.f90.

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

Definition at line 42 of file mo_mhm_constants.f90.

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

Definition at line 23 of file mo_mhm_constants.f90.

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

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

Definition at line 39 of file mo_mhm_constants.f90.

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

Definition at line 33 of file mo_mhm_constants.f90.

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

Definition at line 26 of file mo_mhm_constants.f90.

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

Definition at line 27 of file mo_mhm_constants.f90.

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

Definition at line 28 of file mo_mhm_constants.f90.

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

Definition at line 29 of file mo_mhm_constants.f90.

15.91.2.23 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 51 of file mo_mhm_constants.f90.

15.91.2.24 real(dp), parameter, public mo_mhm_constants::stboltzmann = 5.67e-08_dp

Stefan-Boltzmann constant [W m^-2 K^-4].

Definition at line 37 of file mo_mhm_constants.f90.

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

Tetens's formula to calculate saturated vapour pressure.

Definition at line 47 of file mo_mhm_constants.f90.

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

Definition at line 48 of file mo_mhm_constants.f90.

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

Definition at line 49 of file mo_mhm_constants.f90.

The documentation for this module was generated from the following file:

- mo_mhm_constants.f90

15.92 mo_mhm_eval Module Reference

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

Public Member Functions

- subroutine, public mhm_eval (parameterset, runoff, sm_opti, basin_avg_tws, neutrons_opti, et_opti)

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

15.92.1 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

Definition at line 13 of file mo_mhm_eval.f90.

15.92.2 Member Function/Subroutine Documentation

15.92.2.1 subroutine, public mo_mhm_eval::mhm_eval (real(dp), dimension(:), intent(in) *parameterset,* real(dp), dimension(:, :), intent(out), optional, allocatable *runoff,* real(dp), dimension(:, :), intent(out), optional, allocatable *sm_opti,* real(dp), dimension(:, :), intent(out), optional, allocatable *basin_avg_tws,* real(dp), dimension(:, :), intent(out), optional, allocatable *neutrons_opti,* real(dp), dimension(:, :), intent(out), optional, allocatable *et_opti*)

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

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

Parameters

in	*real(dp), dimension(:) :: parameterset*	a set of global parameter (gamma) to run mHM, DIMENSION [no. of global↩ _Parameters]
out	*real(dp), dimension(:, :), optional :: runoff*	returns runoff time series, DIMENSION [nTimeSteps, nGaugesTotal]
out	*real(dp), dimension(:, :), optional :: sm_opti*	returns soil moisture time series for all grid cells (of multiple basins concatenated),DIMENSION [nCells, nTimeSteps]
out	*real(dp), dimension(:, :), optional :: basin_avg_tws*	returns basin averaged total water storage time series, DIMENSION [nTime↩ Steps, nBasins]
out	*real(dp), dimension(:, :), optional :: neutrons_opti*	dim1=ncells, dim2=time

out	*real(dp), dimension(:, :), optional :: et_opti*	returns evapotranspiration time series for all grid cells (of multiple basins concatenated),DIMENSION [nCells, nTimeSteps]

Authors

Juliane Mai, Rohini Kumar

Date

Feb 2013

Definition at line 83 of file mo_mhm_eval.f90.

References mo_mrm_river_head::avg_and_write_timestep(), mo_mrm_river_head::calc_river_head(), mo_julian↩::caldat(), mo_julian::julday(), mo_runoff::l1_total_runoff(), mo_message::message(), mo_mhm::mhm(), mo_↩mrm_restart::mrm_read_restart_states(), mo_mrm_routing::mrm_routing(), mo_mrm_mpr::mrm_update_param(), mo_mrm_write::mrm_write_output_fluxes(), mo_write_fluxes_states::newoutputdataset(), mo_meteo_forcings↩::prepare_meteo_forcings_data(), mo_restart::read_restart_states(), mo_init_states::variables_default_init(), and mo_mrm_init::variables_default_init_routing().

Referenced by mhm_driver().

Here is the call graph for this function:

Here is the caller graph for this function:

The documentation for this module was generated from the following file:

- mo_mhm_eval.f90

15.93 mo_mhm_read_config Module Reference

Reading of main model configurations.

Public Member Functions

- subroutine, public mhm_read_config (file_namelist, unamelist)

  Read main configurations for mHM.

15.93.1 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

Definition at line 15 of file mo_mhm_read_config.f90.

15.93.2 Member Function/Subroutine Documentation

15.93.2.1 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_nml::close_nml(), mo_message::message(), mo_nml::open_nml(), and mo_nml::position_nml().

Referenced by mhm_driver().

Here is the call graph for this function:

Here is the caller graph for this function:

The documentation for this module was generated from the following file:

- mo_mhm_read_config.f90

15.94 mo_moment Module Reference

Data Types

- interface absdev

- interface average
- interface central_moment
- interface central_moment_var
- interface correlation
- interface covariance
- interface kurtosis
- interface mean
- interface mixed_central_moment
- interface mixed_central_moment_var
- interface moment
- interface skewness
- interface stddev
- interface variance

Private Member Functions

- real(dp) function absdev_dp (dat, mask)
- real(sp) function absdev_sp (dat, mask)
- real(dp) function average_dp (dat, mask)
- real(sp) function average_sp (dat, mask)
- real(dp) function central_moment_dp (x, r, mask)
- real(sp) function central_moment_sp (x, r, mask)
- real(dp) function central_moment_var_dp (x, r, mask)
- real(sp) function central_moment_var_sp (x, r, mask)
- real(dp) function correlation_dp (x, y, mask)
- real(sp) function correlation_sp (x, y, mask)
- real(dp) function covariance_dp (x, y, mask)
- real(sp) function covariance_sp (x, y, mask)
- real(dp) function kurtosis_dp (dat, mask)
- real(sp) function kurtosis_sp (dat, mask)
- real(dp) function mean_dp (dat, mask)
- real(sp) function mean_sp (dat, mask)
- real(dp) function mixed_central_moment_dp (x, y, r, s, mask)
- real(sp) function mixed_central_moment_sp (x, y, r, s, mask)
- real(dp) function mixed_central_moment_var_dp (x, y, r, s, mask)
- real(sp) function mixed_central_moment_var_sp (x, y, r, s, mask)
- subroutine moment_dp (dat, average, variance, skewness, kurtosis, mean, stddev, absdev, mask)
- subroutine moment_sp (dat, average, variance, skewness, kurtosis, mean, stddev, absdev, mask)
- real(dp) function stddev_dp (dat, mask)
- real(sp) function stddev_sp (dat, mask)
- real(dp) function skewness_dp (dat, mask)
- real(sp) function skewness_sp (dat, mask)
- real(dp) function variance_dp (dat, mask)
- real(sp) function variance_sp (dat, mask)

15.94.1 Detailed Description

Definition at line 1 of file mo_moment.f90.

15.94.2 Member Function/Subroutine Documentation

15.94.2.1 real(dp) function mo_moment::absdev_dp (real(dp), dimension(:), intent(in) *dat,* logical, dimension(:), intent(in), optional *mask*) [private]

Definition at line 849 of file mo_moment.f90.

References mo_moment::absdev::absdev_dp().

Here is the call graph for this function:

15.94.2.2 real(sp) function mo_moment::absdev_sp (real(sp), dimension(:), intent(in) *dat,* logical, dimension(:), intent(in), optional *mask*) [private]

Definition at line 880 of file mo_moment.f90.

References mo_moment::absdev::absdev_sp().

Here is the call graph for this function:

15.94.2.3 real(dp) function mo_moment::average_dp (real(dp), dimension(:), intent(in) *dat,* logical, dimension(:), intent(in), optional *mask*) [private]

Definition at line 912 of file mo_moment.f90.

References mo_moment::average::average_dp().

Here is the call graph for this function:

15.94.2.4 real(sp) function mo_moment::average_sp (real(sp), dimension(:), intent(in) *dat*, logical, dimension(:), intent(in), optional *mask*) [private]

Definition at line 938 of file mo_moment.f90.

References mo_moment::average::average_sp().

Here is the call graph for this function:

15.94.2.5 real(dp) function mo_moment::central_moment_dp (real(dp), dimension(:), intent(in) *x*, integer(i4), intent(in) *r*, logical, dimension(:), intent(in), optional *mask*) [private]

Definition at line 965 of file mo_moment.f90.

References mo_moment::central_moment::central_moment_dp().

Here is the call graph for this function:

15.94.2.6 real(sp) function mo_moment::central_moment_sp (real(sp), dimension(:), intent(in) *x*, integer(i4), intent(in) *r*, logical, dimension(:), intent(in), optional *mask*) [private]

Definition at line 1004 of file mo_moment.f90.

References mo_moment::central_moment::central_moment_sp().

Here is the call graph for this function:

15.94.2.7 real(dp) function mo_moment::central_moment_var_dp (real(dp), dimension(:), intent(in) *x*, integer(i4), intent(in) *r*, logical, dimension(:), intent(in), optional *mask*) [private]

Definition at line 1044 of file mo_moment.f90.

References mo_moment::central_moment_var::central_moment_var_dp().

Here is the call graph for this function:

15.94.2.8 real(sp) function mo_moment::central_moment_var_sp (real(sp), dimension(:), intent(in) *x*, integer(i4), intent(in) *r*, logical, dimension(:), intent(in), optional *mask*) [private]

Definition at line 1082 of file mo_moment.f90.

References mo_moment::central_moment_var::central_moment_var_sp().

Here is the call graph for this function:

15.94.2.9 real(dp) function mo_moment::correlation_dp (real(dp), dimension(:), intent(in) *x*, real(dp), dimension(:), intent(in) *y*, logical, dimension(:), intent(in), optional *mask*) [private]

Definition at line 1121 of file mo_moment.f90.

References mo_moment::correlation::correlation_dp().

Here is the call graph for this function:

15.94.2.10 real(sp) function mo_moment::correlation_sp (real(sp), dimension(:), intent(in) *x*, real(sp), dimension(:), intent(in) *y*, logical, dimension(:), intent(in), optional *mask*) [private]

Definition at line 1156 of file mo_moment.f90.

References mo_moment::correlation::correlation_sp().

Here is the call graph for this function:

15.94.2.11 real(dp) function mo_moment::covariance_dp (real(dp), dimension(:), intent(in) *x*, real(dp), dimension(:), intent(in) *y*, logical, dimension(:), intent(in), optional *mask*) [private]

Definition at line 1192 of file mo_moment.f90.

References mo_moment::covariance::covariance_dp().

Here is the call graph for this function:

15.94.2.12 real(sp) function mo_moment::covariance_sp (real(sp), dimension(:), intent(in) *x*, real(sp), dimension(:), intent(in) *y*, logical, dimension(:), intent(in), optional *mask*) [private]

Definition at line 1224 of file mo_moment.f90.

References mo_moment::covariance::covariance_sp().

Here is the call graph for this function:

15.94.2.13 real(dp) function mo_moment::kurtosis_dp (real(dp), dimension(:), intent(in) *dat*, logical, dimension(:), intent(in), optional *mask*) [private]

Definition at line 1257 of file mo_moment.f90.

References mo_moment::kurtosis::kurtosis_dp().

Here is the call graph for this function:

15.94.2.14 real(sp) function mo_moment::kurtosis_sp (real(sp), dimension(:), intent(in) *dat,* logical, dimension(:), intent(in), optional *mask*) [private]

Definition at line 1298 of file mo_moment.f90.

References mo_moment::kurtosis::kurtosis_sp().

Here is the call graph for this function:

15.94.2.15 real(dp) function mo_moment::mean_dp (real(dp), dimension(:), intent(in) *dat,* logical, dimension(:), intent(in), optional *mask*) [private]

Definition at line 1340 of file mo_moment.f90.

References mo_template::mean::mean_dp().

Here is the call graph for this function:

15.94.2.16 real(sp) function mo_moment::mean_sp (real(sp), dimension(:), intent(in) *dat,* logical, dimension(:), intent(in), optional *mask*) [private]

Definition at line 1367 of file mo_moment.f90.

References mo_template::mean::mean_sp().

Here is the call graph for this function:

15.94.2.17 real(dp) function mo_moment::mixed_central_moment_dp (real(dp), dimension(:), intent(in) *x*, real(dp), dimension(:), intent(in) *y*, integer(i4), intent(in) *r*, integer(i4), intent(in) *s*, logical, dimension(:), intent(in), optional *mask*) [private]

Definition at line 1395 of file mo_moment.f90.

References mo_moment::mixed_central_moment::mixed_central_moment_dp().

Here is the call graph for this function:

15.94.2.18 real(sp) function mo_moment::mixed_central_moment_sp (real(sp), dimension(:), intent(in) *x*, real(sp), dimension(:), intent(in) *y*, integer(i4), intent(in) *r*, integer(i4), intent(in) *s*, logical, dimension(:), intent(in), optional *mask*) [private]

Definition at line 1445 of file mo_moment.f90.

References mo_moment::mixed_central_moment::mixed_central_moment_sp().

Here is the call graph for this function:

15.94.2.19 real(dp) function mo_moment::mixed_central_moment_var_dp (real(dp), dimension(:), intent(in) *x,* real(dp), dimension(:), intent(in) *y,* integer(i4), intent(in) *r,* integer(i4), intent(in) *s,* logical, dimension(:), intent(in), optional *mask*) [private]

Definition at line 1496 of file mo_moment.f90.

References mo_moment::mixed_central_moment_var::mixed_central_moment_var_dp().

Here is the call graph for this function:

15.94.2.20 real(sp) function mo_moment::mixed_central_moment_var_sp (real(sp), dimension(:), intent(in) *x,* real(sp), dimension(:), intent(in) *y,* integer(i4), intent(in) *r,* integer(i4), intent(in) *s,* logical, dimension(:), intent(in), optional *mask*) [private]

Definition at line 1542 of file mo_moment.f90.

References mo_moment::mixed_central_moment_var::mixed_central_moment_var_sp().

Here is the call graph for this function:

15.94.2.21 subroutine mo_moment::moment_dp (real(dp), dimension(:), intent(in) *dat,* real(dp), intent(out), optional *average,* real(dp), intent(out), optional *variance,* real(dp), intent(out), optional *skewness,* real(dp), intent(out), optional *kurtosis,* real(dp), intent(out), optional *mean,* real(dp), intent(out), optional *stddev,* real(dp), intent(out), optional *absdev,* logical, dimension(:), intent(in), optional *mask*) [private]

Definition at line 1589 of file mo_moment.f90.

15.94.2.22 subroutine mo_moment::moment_sp (real(sp), dimension(:), intent(in) *dat,* real(sp), intent(out), optional *average,* real(sp), intent(out), optional *variance,* real(sp), intent(out), optional *skewness,* real(sp), intent(out), optional *kurtosis,* real(sp), intent(out), optional *mean,* real(sp), intent(out), optional *stddev,* real(sp), intent(out), optional *absdev,* logical, dimension(:), intent(in), optional *mask*) [private]

Definition at line 1659 of file mo_moment.f90.

15.94.2.23 real(dp) function mo_moment::skewness_dp (real(dp), dimension(:), intent(in) *dat,* logical, dimension(:), intent(in), optional *mask*) [private]

Definition at line 1805 of file mo_moment.f90.

References mo_moment::skewness::skewness_dp().

Here is the call graph for this function:

15.94.2.24 real(sp) function mo_moment::skewness_sp (real(sp), dimension(:), intent(in) *dat,* logical, dimension(:), intent(in), optional *mask*) [private]

Definition at line 1847 of file mo_moment.f90.

References mo_moment::skewness::skewness_sp().

Here is the call graph for this function:

15.94.2.25 real(dp) function mo_moment::stddev_dp (real(dp), dimension(:), intent(in) *dat,* logical, dimension(:), intent(in), optional *mask*) [private]

Definition at line 1730 of file mo_moment.f90.

References mo_moment::stddev::stddev_dp().

Here is the call graph for this function:

15.94.2.26 real(sp) function mo_moment::stddev_sp (real(sp), dimension(:), intent(in) *dat,* logical, dimension(:), intent(in), optional *mask*) [private]

Definition at line 1767 of file mo_moment.f90.

References mo_moment::stddev::stddev_sp().

Here is the call graph for this function:

15.94.2.27 real(dp) function mo_moment::variance_dp (real(dp), dimension(:), intent(in) *dat,* logical, dimension(:), intent(in), optional *mask*) [private]

Definition at line 1890 of file mo_moment.f90.

References mo_moment::variance::variance_dp().

Here is the call graph for this function:

15.94.2.28 real(sp) function mo_moment::variance_sp (real(sp), dimension(:), intent(in) *dat*, logical, dimension(:), intent(in), optional *mask*) [private]

Definition at line 1926 of file mo_moment.f90.

References mo_moment::variance::variance_sp().

Here is the call graph for this function:

The documentation for this module was generated from the following file:

- mo_moment.f90

15.95 mo_mrm_constants Module Reference

Provides mRM specific constants.

Collaboration diagram for mo_mrm_constants:

Public Attributes

- integer(i4), parameter, public noutflxstate = 1_i4
- integer(i4), parameter, public nroutingstates = 2
- integer(i4), parameter, public maxnogauges = 50_i4
- real(dp), parameter, public rout_space_weight = 0._dp
- real(dp), parameter, public deltah = 5.000_dp

- real(dp), dimension(19), parameter given_ts = (/ 60._dp, 120._dp, 180._dp, 240._dp, 300._dp, 360._dp, 600._dp, 720._dp, 900._dp, 1200._dp, 1800._dp, 3600._dp, 7200._dp, 10800._dp, 14400._dp, 21600._dp, 28800._dp, 43200._dp, 86400._dp/)

15.95.1 Detailed Description

Provides mRM specific constants.

Provides mRM specific constants such as flood plain elevation.

Authors

Stephan Thober

Date

Aug 2015

Definition at line 14 of file mo_mrm_constants.f90.

15.95.2 Member Data Documentation

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

Definition at line 37 of file mo_mrm_constants.f90.

15.95.2.2 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.

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

Definition at line 27 of file mo_mrm_constants.f90.

15.95.2.4 integer(i4), parameter, public mo_mrm_constants::noutflxstate = 1_i4

Definition at line 18 of file mo_mrm_constants.f90.

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

Definition at line 20 of file mo_mrm_constants.f90.

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

Definition at line 32 of file mo_mrm_constants.f90.

The documentation for this module was generated from the following file:

- mo_mrm_constants.f90

15.96 mo_mrm_eval Module Reference

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

Public Member Functions

- subroutine, public mrm_eval (parameterset, runoff, sm_opti, basin_avg_tws, neutrons_opti, et_opti)

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

15.96.1 Detailed Description

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

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

Authors

Stephan Thober

Date

Sep 2015

Definition at line 13 of file mo_mrm_eval.f90.

15.96.2 Member Function/Subroutine Documentation

15.96.2.1 subroutine, public mo_mrm_eval::mrm_eval (real(dp), dimension(:), intent(in) *parameterset,* real(dp), dimension(:, :), intent(out), optional, allocatable *runoff,* real(dp), dimension(:, :), intent(out), optional, allocatable *sm_opti,* real(dp), dimension(:, :), intent(out), optional, allocatable *basin_avg_tws,* real(dp), dimension(:, :), intent(out), optional, allocatable *neutrons_opti,* real(dp), dimension(:, :), intent(out), optional, allocatable *et_opti*)

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

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

Parameters

in	*real(dp), dimension(:) :: parameterset*	a set of global parameter (gamma) to run mHM, DIMENSION [no. of global↩ _Parameters]
out	*real(dp), dimension(:, :), optional :: runoff*	returns runoff time series, DIMENSION [nTimeSteps, nGaugesTotal]
out	*real(dp), dimension(:, :), optional :: sm_opti*	dim1=ncells, dim2=time
out	*real(dp), dimension(:, :), optional :: basin_avg_tws*	dim1=time dim2=nBasins

out	*real(dp), dimension(:, :), optional :: neutrons_opti*	dim1=ncells, dim2=time
out	*real(dp), dimension(:, :), optional :: et_opti*	dim1=ncells, dim2=time

Authors

Stephan Thober

Date

Sep 2015

Definition at line 52 of file mo_mrm_eval.f90.

References mo_mrm_river_head::avg_and_write_timestep(), mo_mrm_river_head::calc_river_head(), mo_julian↩::caldat(), mo_julian::julday(), mo_message::message(), mo_mrm_restart::mrm_read_restart_states(), mo_mrm↩_routing::mrm_routing(), mo_mrm_mpr::mrm_update_param(), mo_mrm_write::mrm_write_output_fluxes(), and mo_mrm_init::variables_default_init_routing().

Referenced by mrm_driver().

Here is the call graph for this function:

Here is the caller graph for this function:

The documentation for this module was generated from the following file:

- mo_mrm_eval.f90

15.97 mo_mrm_file Module Reference

Provides file names and units for mRM.

Collaboration diagram for mo_mrm_file:

Public Attributes

- character(len=∗), parameter version = '1.1'

  Current mHM model version.
- character(len=∗), parameter version_date = 'Nov 2016'

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

15.97.1 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

Definition at line 13 of file mo_mrm_file.f90.

15.97.2 Member Data Documentation

15.97.2.1 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.

15.97.2.2 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.

15.97.2.3 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.

15.97.2.4 character(len = ∗), parameter mo_mrm_file::file_facc = 'facc.asc'

Definition at line 32 of file mo_mrm_file.f90.

15.97.2.5 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.

15.97.2.6 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.

15.97.2.7 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.

15.97.2.8 character(len = ∗), parameter mo_mrm_file::file_main = 'mrm_driver.f90'

Driver file.

Definition at line 22 of file mo_mrm_file.f90.

15.97.2.9 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.

15.97.2.10 character(len = ∗), parameter mo_mrm_file::file_namelist_mrm = 'mrm.nml'

Namelist file name.

Definition at line 24 of file mo_mrm_file.f90.

15.97.2.11 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.

15.97.2.12 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.

15.97.2.13 character(len = ∗), parameter mo_mrm_file::ncfile_discharge = 'discharge.nc'

file defining optimazation outputs

Definition at line 67 of file mo_mrm_file.f90.

15.97.2.14 integer, parameter mo_mrm_file::uconfig = 68

Unit for file defining mHM's outputs.

Definition at line 60 of file mo_mrm_file.f90.

15.97.2.15 integer, parameter mo_mrm_file::udaily_discharge = 74

Unit for file optimazation outputs.

Definition at line 65 of file mo_mrm_file.f90.

15.97.2.16 integer, parameter mo_mrm_file::udefoutput = 67

Unit for file defining mRM's outputs.

Definition at line 55 of file mo_mrm_file.f90.

15.97.2.17 integer, parameter mo_mrm_file::udischarge = 66

unit for discharge time series

Definition at line 50 of file mo_mrm_file.f90.

15.97.2.18 integer, parameter mo_mrm_file::ufacc = 56

Unit for flow accumulation input data file.

Definition at line 34 of file mo_mrm_file.f90.

15.97.2.19 integer, parameter mo_mrm_file::ufdir = 57

Unit for flow direction input data file.

Definition at line 38 of file mo_mrm_file.f90.

15.97.2.20 integer, parameter mo_mrm_file::ugaugeloc = 62

Unit for gauge location input data file.

Definition at line 47 of file mo_mrm_file.f90.

15.97.2.21 integer, parameter mo_mrm_file::unamelist_mrm = 40

Unit for namelist.

Definition at line 26 of file mo_mrm_file.f90.

15.97.2.22 integer, parameter mo_mrm_file::unamelist_param_mrm = 41

Unit for namelist.

Definition at line 30 of file mo_mrm_file.f90.

15.97.2.23 integer, parameter mo_mrm_file::uslope = 59

Unit for flow direction input data file.

Definition at line 42 of file mo_mrm_file.f90.

15.97.2.24 character(len = ∗), parameter mo_mrm_file::version = '1.1'

Current mHM model version.

Definition at line 18 of file mo_mrm_file.f90.

15.97.2.25 character(len = ∗), parameter mo_mrm_file::version_date = 'Nov 2016'

Time of current mHM model version release.

Definition at line 20 of file mo_mrm_file.f90.

The documentation for this module was generated from the following file:

- mo_mrm_file.f90

15.98 mo_mrm_global_variables Module Reference

Global variables for mRM only.

Collaboration diagram for mo_mrm_global_variables:

Data Types

- type basininfo_mrm
- type gaugingstation

Public Attributes

- logical is_start
- integer(i4) timestep_model_outputs_mrm
- logical, dimension(noutflxstate) outputflxstate_mrm

- character(256), dimension(:), allocatable, public dirgauges
- character(256), dimension(:), allocatable, public dirtotalrunoff
- character(256), public filenametotalrunoff
- character(256), public varnametotalrunoff
- character(256), dimension(:), allocatable, public dirbankfullrunoff
- integer(i4), public ntstepday
- type(grid), dimension(:), allocatable, target, public level11
- type(gridremapper), dimension(:), allocatable, public l0_l11_remap
- type(gridremapper), dimension(:), allocatable, public l1_l11_remap
- real(dp), dimension(:,:), allocatable, public mrm_runoff
- integer(i4), public ngaugestotal
- integer(i4), public ninflowgaugestotal
- integer(i4), public nmeasperday
- type(gaugingstation), public gauge
- type(gaugingstation), public inflowgauge
- type(basininfo_mrm), dimension(:), allocatable, target, public basin_mrm
- integer(i4), dimension(:), allocatable, public l0_gaugeloc
- integer(i4), dimension(:), allocatable, public l0_inflowgaugeloc
- integer(i4), dimension(:), allocatable, public l0_facc
- integer(i4), dimension(:), allocatable, public l0_fdir
- integer(i4), dimension(:), allocatable, public l0_drasc
- integer(i4), dimension(:), allocatable, public l0_dracell
- integer(i4), dimension(:), allocatable, public l0_streamnet
- integer(i4), dimension(:), allocatable, public l0_floodplain
- integer(i4), dimension(:), allocatable, public l0_noutlet
- real(dp), dimension(:), allocatable, public l0_celerity
- integer(i4), dimension(:), allocatable, public l11_l1_id
- real(dp), dimension(:,:), allocatable, public l1_total_runoff_in
- integer(i4), dimension(:,:), allocatable, public l11_cellcoor
- integer(i4), dimension(:), allocatable, public l1_l11_id
- real(dp), dimension(:), allocatable, public l11_areacell
- real(dp), dimension(:), allocatable, public l11_facc

- integer(i4), dimension(:), allocatable, public l11_fdir
- integer(i4), dimension(:), allocatable, public l11_noutlets
- real(dp), dimension(:), allocatable, public l11_celerity
- real(dp), dimension(:), allocatable, public l11_meandering
- real(dp), dimension(:), allocatable, public l11_linkin_facc
- integer(i4), dimension(:), allocatable, public l11_rowout
- integer(i4), dimension(:), allocatable, public l11_colout
- real(dp), dimension(:), allocatable, public l11_qmod
- real(dp), dimension(:), allocatable, public l11_qout
- real(dp), dimension(:,:), allocatable, public l11_qtin
- real(dp), dimension(:,:), allocatable, public l11_qtr
- integer(i4), dimension(:), allocatable, public l11_fromn
- integer(i4), dimension(:), allocatable, public l11_ton
- integer(i4), dimension(:), allocatable, public l11_netperm
- integer(i4), dimension(:), allocatable, public l11_frow
- integer(i4), dimension(:), allocatable, public l11_fcol
- integer(i4), dimension(:), allocatable, public l11_trow
- integer(i4), dimension(:), allocatable, public l11_tcol
- integer(i4), dimension(:), allocatable, public l11_rorder
- integer(i4), dimension(:), allocatable, public l11_label
- logical, dimension(:), allocatable, public l11_sink
- real(dp), dimension(:), allocatable, public l11_length
- real(dp), dimension(:), allocatable, target, public l11_afloodplain
- real(dp), dimension(:), allocatable, public l11_slope
- real(dp), dimension(:,:), allocatable, public l11_nlinkfracfpimp
- real(dp), dimension(:), allocatable, public l11_k
- real(dp), dimension(:), allocatable, public l11_xi
- real(dp), dimension(:), allocatable, public l11_tsrout

- real(dp), dimension(:), allocatable, public l11_c1
- real(dp), dimension(:), allocatable, public l11_c2
- logical gw_coupling
- real(dp), dimension(:), allocatable, public l11_bankfull_runoff_in
- real(dp), dimension(:), allocatable, public l0_channel_depth
- real(dp), dimension(:), allocatable, public l0_channel_elevation
- real(dp), dimension(:), allocatable, public l0_river_head_mon_sum
- real(dp), dimension(:), allocatable, public l0_slope

15.98.1 Detailed Description

Global variables for mRM only.

TODO: add description

Authors

Luis Samaniego, Stephan Thober

Date

Aug 2015

Definition at line 16 of file mo_mrm_global_variables.f90.

15.98.2 Member Data Documentation

15.98.2.1 type(basininfo_mrm), dimension(:), allocatable, target, public mo_mrm_global_variables::basin_mrm

Definition at line 112 of file mo_mrm_global_variables.f90.

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

Definition at line 48 of file mo_mrm_global_variables.f90.

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

Definition at line 44 of file mo_mrm_global_variables.f90.

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

Definition at line 45 of file mo_mrm_global_variables.f90.

15.98.2.5 character(256), public mo_mrm_global_variables::filenametotalrunoff

Definition at line 46 of file mo_mrm_global_variables.f90.

15.98.2.6 type(gaugingstation), public mo_mrm_global_variables::gauge

Definition at line 83 of file mo_mrm_global_variables.f90.

15.98.2.7 logical mo_mrm_global_variables::gw_coupling

Definition at line 211 of file mo_mrm_global_variables.f90.

15.98.2.8 type(gaugingstation), public mo_mrm_global_variables::inflowgauge

Definition at line 84 of file mo_mrm_global_variables.f90.

15.98.2.9 logical mo_mrm_global_variables::is_start

Definition at line 29 of file mo_mrm_global_variables.f90.

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

Definition at line 132 of file mo_mrm_global_variables.f90.

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

Definition at line 215 of file mo_mrm_global_variables.f90.

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

Definition at line 216 of file mo_mrm_global_variables.f90.

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

Definition at line 128 of file mo_mrm_global_variables.f90.

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

Definition at line 126 of file mo_mrm_global_variables.f90.

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

Definition at line 121 of file mo_mrm_global_variables.f90.

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

Definition at line 122 of file mo_mrm_global_variables.f90.

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

Definition at line 130 of file mo_mrm_global_variables.f90.

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

Definition at line 119 of file mo_mrm_global_variables.f90.

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

Definition at line 120 of file mo_mrm_global_variables.f90.

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

Definition at line 60 of file mo_mrm_global_variables.f90.

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

Definition at line 131 of file mo_mrm_global_variables.f90.

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

Definition at line 218 of file mo_mrm_global_variables.f90.

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

Definition at line 219 of file mo_mrm_global_variables.f90.

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

Definition at line 129 of file mo_mrm_global_variables.f90.

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

Definition at line 191 of file mo_mrm_global_variables.f90.

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

Definition at line 154 of file mo_mrm_global_variables.f90.

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

Definition at line 213 of file mo_mrm_global_variables.f90.

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

Definition at line 204 of file mo_mrm_global_variables.f90.

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

Definition at line 205 of file mo_mrm_global_variables.f90.

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

Definition at line 158 of file mo_mrm_global_variables.f90.

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

Definition at line 151 of file mo_mrm_global_variables.f90.

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

Definition at line 167 of file mo_mrm_global_variables.f90.

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

Definition at line 155 of file mo_mrm_global_variables.f90.

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

Definition at line 184 of file mo_mrm_global_variables.f90.

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

Definition at line 156 of file mo_mrm_global_variables.f90.

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

Definition at line 180 of file mo_mrm_global_variables.f90.

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

Definition at line 183 of file mo_mrm_global_variables.f90.

15.98.2.38 real(dp), dimension(:), allocatable, public mo_mrm_global_variables::l11_k

Definition at line 200 of file mo_mrm_global_variables.f90.

15.98.2.39 integer(i4), dimension(:), allocatable, public mo_mrm_global_variables::l11_l1_id

Definition at line 138 of file mo_mrm_global_variables.f90.

15.98.2.40 integer(i4), dimension(:), allocatable, public mo_mrm_global_variables::l11_label

Definition at line 188 of file mo_mrm_global_variables.f90.

15.98.2.41 real(dp), dimension(:), allocatable, public mo_mrm_global_variables::l11_length

Definition at line 190 of file mo_mrm_global_variables.f90.

15.98.2.42 real(dp), dimension(:), allocatable, public mo_mrm_global_variables::l11_linkin_facc

Definition at line 162 of file mo_mrm_global_variables.f90.

15.98.2.43 real(dp), dimension(:), allocatable, public mo_mrm_global_variables::l11_meandering

Definition at line 160 of file mo_mrm_global_variables.f90.

15.98.2.44 integer(i4), dimension(:), allocatable, public mo_mrm_global_variables::l11_netperm

Definition at line 182 of file mo_mrm_global_variables.f90.

15.98.2.45 real(dp), dimension(:, :), allocatable, public mo_mrm_global_variables::l11_nlinkfracfpimp

Definition at line 197 of file mo_mrm_global_variables.f90.

15.98.2.46 integer(i4), dimension(:), allocatable, public mo_mrm_global_variables::l11_noutlets

Definition at line 157 of file mo_mrm_global_variables.f90.

15.98.2.47 real(dp), dimension(:), allocatable, public mo_mrm_global_variables::l11_qmod

Definition at line 175 of file mo_mrm_global_variables.f90.

15.98.2.48 real(dp), dimension(:), allocatable, public mo_mrm_global_variables::l11_qout

Definition at line 176 of file mo_mrm_global_variables.f90.

15.98.2.49 real(dp), dimension(:, :), allocatable, public mo_mrm_global_variables::l11_qtin

Definition at line 177 of file mo_mrm_global_variables.f90.

15.98.2.50 real(dp), dimension(:, :), allocatable, public mo_mrm_global_variables::l11_qtr

Definition at line 178 of file mo_mrm_global_variables.f90.

15.98.2.51 integer(i4), dimension(:), allocatable, public mo_mrm_global_variables::l11_rorder

Definition at line 187 of file mo_mrm_global_variables.f90.

15.98.2.52 integer(i4), dimension(:), allocatable, public mo_mrm_global_variables::l11_rowout

Definition at line 166 of file mo_mrm_global_variables.f90.

15.98.2.53 logical, dimension(:), allocatable, public mo_mrm_global_variables::l11_sink

Definition at line 189 of file mo_mrm_global_variables.f90.

15.98.2.54 real(dp), dimension(:), allocatable, public mo_mrm_global_variables::l11_slope

Definition at line 193 of file mo_mrm_global_variables.f90.

15.98.2.55 integer(i4), dimension(:), allocatable, public mo_mrm_global_variables::l11_tcol

Definition at line 186 of file mo_mrm_global_variables.f90.

15.98.2.56 integer(i4), dimension(:), allocatable, public mo_mrm_global_variables::l11_ton

Definition at line 181 of file mo_mrm_global_variables.f90.

15.98.2.57 integer(i4), dimension(:), allocatable, public mo_mrm_global_variables::l11_trow

Definition at line 185 of file mo_mrm_global_variables.f90.

15.98.2.58 real(dp), dimension(:), allocatable, public mo_mrm_global_variables::l11_tsrout

Definition at line 203 of file mo_mrm_global_variables.f90.

15.98.2.59 real(dp), dimension(:), allocatable, public mo_mrm_global_variables::l11_xi

Definition at line 201 of file mo_mrm_global_variables.f90.

15.98.2.60 integer(i4), dimension(:), allocatable, public mo_mrm_global_variables::l1_l11_id

Definition at line 153 of file mo_mrm_global_variables.f90.

15.98.2.61 type(gridremapper), dimension(:), allocatable, public mo_mrm_global_variables::l1_l11_remap

Definition at line 61 of file mo_mrm_global_variables.f90.

15.98.2.62 real(dp), dimension(:, :), allocatable, public mo_mrm_global_variables::l1_total_runoff_in

Definition at line 144 of file mo_mrm_global_variables.f90.

15.98.2.63 type(grid), dimension(:), allocatable, target, public mo_mrm_global_variables::level11

Definition at line 59 of file mo_mrm_global_variables.f90.

15.98.2.64 real(dp), dimension(:, :), allocatable, public mo_mrm_global_variables::mrm_runoff

Definition at line 67 of file mo_mrm_global_variables.f90.

15.98.2.65 integer(i4), public mo_mrm_global_variables::ngaugestotal

Definition at line 72 of file mo_mrm_global_variables.f90.

15.98.2.66 integer(i4), public mo_mrm_global_variables::ninflowgaugestotal

Definition at line 73 of file mo_mrm_global_variables.f90.

15.98.2.67 integer(i4), public mo_mrm_global_variables::nmeasperday

Definition at line 74 of file mo_mrm_global_variables.f90.

15.98.2.68 integer(i4), public mo_mrm_global_variables::ntstepday

Definition at line 53 of file mo_mrm_global_variables.f90.

15.98.2.69 logical, dimension(noutflxstate) mo_mrm_global_variables::outputflxstate_mrm

Definition at line 37 of file mo_mrm_global_variables.f90.

15.98.2.70 integer(i4) mo_mrm_global_variables::timestep_model_outputs_mrm

Definition at line 35 of file mo_mrm_global_variables.f90.

15.98.2.71 character(256), public mo_mrm_global_variables::varnametotalrunoff

Definition at line 47 of file mo_mrm_global_variables.f90.

The documentation for this module was generated from the following file:

- mo_mrm_global_variables.f90

15.99 mo_mrm_init Module Reference

Wrapper for initializing Routing.

Public Member Functions

- subroutine, public mrm_init (file_namelist, unamelist, file_namelist_param, unamelist_param)

  Initialize all mRM variables at all levels (i.e., L0, L1, and L11).
- subroutine print_startup_message (file_namelist, file_namelist_param)

  TODO: add description.
- subroutine config_output

  TODO: add description.
- subroutine, public variables_default_init_routing

  Default initalization mRM related L11 variables.
- subroutine l0_check_input_routing (L0Basin_iBasin)

  TODO: add description.
- subroutine variables_alloc_routing (iBasin)

  TODO: add description.

15.99.1 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

Definition at line 13 of file mo_mrm_init.f90.

15.99.2 Member Function/Subroutine Documentation

15.99.2.1 subroutine mo_mrm_init::config_output ( )

TODO: add description.

TODO: add description

Authors

Robert Schweppe

Date

Jun 2018

Definition at line 354 of file mo_mrm_init.f90.

References mo_message::message().

Referenced by mrm_init().

Here is the call graph for this function:

Here is the caller graph for this function:

15.99.2.2 subroutine mo_mrm_init::l0_check_input_routing (integer(i4), intent(in) *L0Basin_iBasin*)

TODO: add description.

TODO: add description

Parameters

in	*integer(i4) :: L0Basin_iBasin*	

Authors

Robert Schweppe

Date

Jun 2018

Definition at line 488 of file mo_mrm_init.f90.

References mo_message::message().

Referenced by mrm_init().

Here is the call graph for this function:

Here is the caller graph for this function:

15.99.2.3 subroutine, public mo_mrm_init::mrm_init (character(∗), intent(in) *file_namelist,* integer, intent(in) *unamelist,* character(∗), intent(in) *file_namelist_param,* integer, intent(in) *unamelist_param*)

Initialize all mRM variables at all levels (i.e., L0, L1, and L11).

Initialize all mRM variables at all levels (i.e., L0, L1, and L11) either with default values or with values from restart file. The L0 mask (L0_mask), L0 elevation (L0_elev), and L0 land cover (L0_LCover) can be provided as optional variables to save memory because these variable will then not be read in again.

Parameters

in	*character(∗) :: file_namelist, file_namelist_param*	
in	*integer :: unamelist, unamelist_param*	
in	*character(∗) :: file_namelist, file_namelist_param*	
in	*integer :: unamelist, unamelist_param*	

Authors

Stephan Thober

Date

Aug 2015

Definition at line 62 of file mo_mrm_init.f90.

References mo_mrm_river_head::calc_channel_elevation(), mo_common_read_config::common_read_config(), config_output(), mo_mrm_river_head::create_output(), mo_grid::init_lowres_level(), mo_mrm_river_head::init_masked_zeros_l0(), l0_check_input_routing(), mo_grid::l0_grid_setup(), mo_mrm_net_startup::l11_flow_direction(), mo_mrm_net_startup::l11_fraction_sealed_floodplain(), mo_mrm_net_startup::l11_l1_mapping(), mo_mrm_net_startup::l11_link_location(), mo_mrm_net_startup::l11_routing_order(), mo_mrm_net_startup::l11_set_drain_outlet_gauges(), mo_mrm_net_startup::l11_set_network_topology(), mo_mrm_net_startup::l11_stream_features(), mo_message::message(), mo_mrm_mpr::mrm_init_param(), mo_mrm_read_data::mrm_read_bankfull_runoff(), mo_mrm_read_config::mrm_read_config(), mo_mrm_read_data::mrm_read_discharge(),

mo_mrm_read_data::mrm_read_l0_data(), mo_mrm_restart::mrm_read_restart_config(), mo_mrm_read_data↩ ::mrm_read_total_runoff(), print_startup_message(), mo_common_restart::read_grid_info(), mo_read_latlon↩ ::read_latlon(), mo_grid::set_basin_indices(), and variables_alloc_routing().

Referenced by mhm_driver(), and mrm_driver().

Here is the call graph for this function:

Here is the caller graph for this function:

15.99.2.4 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 293 of file mo_mrm_init.f90.

References mo_message::message().

Referenced by mrm_init().

Here is the call graph for this function:

Here is the caller graph for this function:

15.99.2.5 subroutine mo_mrm_init::variables_alloc_routing (integer(i4), intent(in) *iBasin*)

TODO: add description.

TODO: add description

Parameters

in	*integer(i4) :: iBasin*	

Authors

Robert Schweppe

Date

Jun 2018

Definition at line 546 of file mo_mrm_init.f90.

Referenced by mrm_init().

Here is the caller graph for this function:

15.99.2.6 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 436 of file mo_mrm_init.f90.

Referenced by mo_mhm_eval::mhm_eval(), and mo_mrm_eval::mrm_eval().

Here is the caller graph for this function:

The documentation for this module was generated from the following file:

- mo_mrm_init.f90

15.100 mo_mrm_mpr Module Reference

Perform Multiscale Parameter Regionalization on Routing Parameters.

Public Member Functions

- subroutine, public reg_rout (param, length, slope, fFPimp, TS, C1, C2)

  Regionalized routing.
- subroutine, public mrm_init_param (iBasin, param)

  TODO: add description.
- subroutine, public mrm_update_param (iBasin, param)

  TODO: add description.

15.100.1 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

Definition at line 15 of file mo_mrm_mpr.f90.

15.100.2 Member Function/Subroutine Documentation

15.100.2.1 subroutine, public mo_mrm_mpr::mrm_init_param (integer(i4), intent(in) *iBasin,* real(dp), dimension(:), intent(in) *param*)

TODO: add description.

TODO: add description

Parameters

in	*integer(i4) :: iBasin*	Basin number
in	*real(dp), dimension(:) :: param*	input parameter (param(1) is celerity in m/s)

Authors

Stephan Thober, Matthias Kelbling

Date

Jun 2018

Definition at line 144 of file mo_mrm_mpr.f90.

References mo_mrm_net_startup::l11_calc_celerity(), mo_message::message(), and mrm_update_param().

Referenced by mo_mrm_init::mrm_init().

Here is the call graph for this function:

Here is the caller graph for this function:

15.100.2.2 subroutine, public mo_mrm_mpr::mrm_update_param (integer(i4), intent(in) *iBasin,* real(dp), dimension(1), intent(in) *param*)

TODO: add description.

TODO: add description

Parameters

in	*integer(i4) :: iBasin*	Basin number
in	*real(dp), dimension(1) :: param*	celerity parameter [m s-1]

Authors

Stehpan Thober, Matthias Kelbling

Date

Jun 2018

Definition at line 241 of file mo_mrm_mpr.f90.

References mo_mrm_net_startup::l11_calc_celerity(), and mo_message::message().

Referenced by mo_mhm_eval::mhm_eval(), mo_mrm_eval::mrm_eval(), and mrm_init_param().

Here is the call graph for this function:

Here is the caller graph for this function:

15.100.2.3 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:

The documentation for this module was generated from the following file:

- mo_mrm_mpr.f90

15.101 mo_mrm_net_startup Module Reference

Startup drainage network for mHM.

Public Member Functions

- subroutine, public l11_l1_mapping (iBasin)

  TODO: add description.
- subroutine, public l11_flow_direction (iBasin)

  Determine the flow direction of the upscaled river network at level L11.
- subroutine, public l11_set_network_topology (iBasin)

  Set network topology.
- subroutine, public l11_routing_order (iBasin)

  Find routing order, headwater cells and sink.
- subroutine, public l11_link_location (iBasin)

  Estimate the LO (row,col) location for each routing link at level L11.
- subroutine, public l11_set_drain_outlet_gauges (iBasin)

  Draining cell identification and Set gauging node.
- subroutine, public l11_stream_features (iBasin)

  Stream features (stream network and floodplain)
- subroutine, public l11_fraction_sealed_floodplain (LCClassImp, do_init)

  Fraction of the flood plain with impervious cover.
- subroutine moveup (elev0, fDir0, fi, fj, ss, nn)

  TODO: add description.
- subroutine movedownonecell (fDir, iRow, jCol)

  TODO: add description.
- subroutine celllength (iBasin, fDir, iRow, jCol, iCoorSystem, length)

  TODO: add description.
- subroutine, public get_distance_two_lat_lon_points (lat1, long1, lat2, long2, distance_out)

  estimate distance in [m] between two points in a lat-lon
- subroutine, public l11_flow_accumulation (iBasin)

Calculates L11 flow accumulation per grid cell.

- subroutine, public l11_calc_celerity (iBasin, param)

  L11 celerity based on L0 elevation and L0 fAcc.

15.101.1 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

Definition at line 21 of file mo_mrm_net_startup.f90.

15.101.2 Member Function/Subroutine Documentation

15.101.2.1 subroutine mo_mrm_net_startup::celllength (integer(i4), intent(in) *iBasin,* integer(i4), intent(in) *fDir,* integer(i4), intent(in) *iRow,* integer(i4), intent(in) *jCol,* integer(i4), intent(in) *iCoorSystem,* real(dp), intent(out) *length*)

TODO: add description.

TODO: add description

Parameters

in	*integer(i4) :: iBasin*	
in	*integer(i4) :: fDir*	
in	*integer(i4) :: iRow*	
in	*integer(i4) :: jCol*	
in	*integer(i4) :: iCoorSystem*	
out	*real(dp) :: length*	

Authors

Robert Schweppe

Date

Jun 2018

Definition at line 1817 of file mo_mrm_net_startup.f90.

References get_distance_two_lat_lon_points(), 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:

15.101.2.2 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 1911 of file mo_mrm_net_startup.f90.

Referenced by celllength().

Here is the caller graph for this function:

15.101.2.3 subroutine, public mo_mrm_net_startup::l11_calc_celerity (integer(i4), intent(in) *iBasin,* real(dp), dimension(:), intent(in) *param*)

L11 celerity based on L0 elevation and L0 fAcc.

L11 celerity based on L0 elevation and L0 fAcc

Parameters

in		author Matthias Kelbling

Date

Oct 2017

Definition at line 2221 of file mo_mrm_net_startup.f90.

References movedownonecell().

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:

15.101.2.4 subroutine, public mo_mrm_net_startup::l11_flow_accumulation (integer(i4), intent(in) *iBasin*)

Calculates L11 flow accumulation per grid cell.

Calculates L11 flow accumulation per grid cell using L11_fDir and L11_cellarea. L11_flow_accumulation contains the recursiv subroutine calculate_L11_flow_accumulation iBasin

Author

Matthias Kelbling

Date

Aug 2017

Definition at line 2007 of file mo_mrm_net_startup.f90.

References calculate_l11_flow_accumulation().

Here is the call graph for this function:

15.101.2.5 subroutine, public mo_mrm_net_startup::l11_flow_direction (integer(i4), intent(in) *iBasin*)

Determine the flow direction of the upscaled river network at level L11.

The hydrographs generated at each cell are routed through the drainage network at level-11 towards their outlets. The drainage network at level-11 is conceptualized as a graph whose nodes are hypothetically located at the center of each grid cell connected by links that represent the river reaches. The flow direction of a link correspond to the direction towards a neighboring cell in which the net flow accumulation (outflows minus inflows) attains its maximum value. The net flow accumulation across a cell's boundary at level-11 is estimated based on flow direction and flow accumulation obtained at level-0 (Routing Network"). Note: level-1 denotes the modeling level, whereas level-L11 is at least as coarse as level-1. Experience has shown that routing can be done at a coarser resolution as level-1, hence the level-11 was introduced.

Figure 15.1: Upscaling routing network from L0 to L1 (or L11)

The left panel depicts a schematic derivation of a drainage network at the level-11 based on level-0 flow direction and flow accumulation. The dotted line circle denotes the point with the highest flow accumulation within a grid cell. The topology of a tipical drainage routing network at level-11 is shown in the right panel. Gray color areas denote the flood plains estimated in mo_init_mrm, where the network upscaling is also carried out. For the sake of simplicity, it is assumed that all runoff leaving a given cell would exit through a major direction. Note that multiple outlets can exist within the modelling domain. If a variable is added or removed here, then it also has to be added or removed in the subroutine L11_config_set in module mo_restart and in the subroutine set_L11_config in module mo_set_netcdf_restart ADDITIONAL INFORMATION L11_flow_direction

Parameters

in	*integer(i4) :: iBasin*	Basin Id

Authors

Luis Samaniego

Date

Dec 2005

Definition at line 221 of file mo_mrm_net_startup.f90.

References mo_message::message(), and movedownonecell().

Referenced by mo_mrm_init::mrm_init().

Here is the call graph for this function:

Here is the caller graph for this function:

15.101.2.6 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. [**?**] found out that this fraction is one of the statistically significant predictor variables of peak discharge in mesoscale basins. If a variable is added or removed here, then it also has to be added or removed in the subroutine L11_config_set in module mo_restart and in the subroutine set_L11_↩ config in module mo_set_netcdf_restart

Parameters

in	*integer(i4) :: LCClassImp*	Impervious land cover class Id, e.g. = 2 (old code)
in	*logical :: do_init*	

Authors

Luis Samaniego

Date

Dec 2005

Definition at line 1497 of file mo_mrm_net_startup.f90.

Referenced by mo_mrm_init::mrm_init().

Here is the caller graph for this function:

15.101.2.7 subroutine, public mo_mrm_net_startup::l11_l1_mapping (integer(i4), intent(in) *iBasin*)

TODO: add description.

TODO: add description

Parameters

in	*integer(i4) :: iBasin*	basin

Authors

Robert Schweppe

Date

Jun 2018

Definition at line 55 of file mo_mrm_net_startup.f90.

Referenced by mo_mrm_init::mrm_init().

Here is the caller graph for this function:

15.101.2.8 subroutine, public mo_mrm_net_startup::l11_link_location (integer(i4), intent(in) *iBasin*)

Estimate the LO (row,col) location for each routing link at level L11.

If a variable is added or removed here, then it also has to be added or removed in the subroutine L11_config_set in module mo_restart and in the subroutine set_L11_config in module mo_set_netcdf_restart

Parameters

in	*integer(i4) :: iBasin*	Basin Id

Authors

Luis Samaniego

Date

Dec 2005

Definition at line 880 of file mo_mrm_net_startup.f90.

References mo_message::message(), and movedownonecell().

Referenced by mo_mrm_init::mrm_init().

Here is the call graph for this function:

Here is the caller graph for this function:

15.101.2.9 subroutine, public mo_mrm_net_startup::l11_routing_order (integer(i4), intent(in) *iBasin*)

Find routing order, headwater cells and sink.

Find routing order, headwater cells and sink. If a variable is added or removed here, then it also has to be added or removed in the subroutine L11_config_set in module mo_restart and in the subroutine set_L11_config in module mo_set_netcdf_restart

Parameters

in	*integer(i4) :: iBasin*	Basin Id

Authors

Luis Samaniego

Date

Dec 2005

Definition at line 721 of file mo_mrm_net_startup.f90.

Referenced by mo_mrm_init::mrm_init().

Here is the caller graph for this function:

15.101.2.10 subroutine, public mo_mrm_net_startup::l11_set_drain_outlet_gauges (integer(i4), intent(in) *iBasin*)

Draining cell identification and Set gauging node.

Perform the following tasks:

- Draining cell identification (cell at L0 to draining cell outlet at L11).
- Set gauging nodes If a variable is added or removed here, then it also has to be added or removed in the subroutine L11_config_set in module mo_restart and in the subroutine set_L11_config in module mo_set_netcdf_restart

  Parameters

in	*integer(i4) :: iBasin*	Basin Id

  Authors

  Luis Samaniego

  Date

  Dec 2005

Definition at line 1082 of file mo_mrm_net_startup.f90.

References movedownonecell().

Referenced by mo_mrm_init::mrm_init().

Here is the call graph for this function:

Here is the caller graph for this function:

15.101.2.11 subroutine, public mo_mrm_net_startup::l11_set_network_topology (integer(i4), intent(in) *iBasin*)

Set network topology.

Set network topology from and to node for all links at level-11 (Routing Network). If a variable is added or removed here, then it also has to be added or removed in the subroutine L11_config_set in module mo_restart and in the subroutine set_L11_config in module mo_set_netcdf_restart.

Parameters

in	*integer(i4) :: iBasin*	Basin Id

Authors

Luis Samaniego

Date

Dec 2005

Definition at line 623 of file mo_mrm_net_startup.f90.

References movedownonecell().

Referenced by mo_mrm_init::mrm_init().

Here is the call graph for this function:

Here is the caller graph for this function:

15.101.2.12 subroutine, public mo_mrm_net_startup::l11_stream_features (integer(i4), intent(in) *iBasin*)

Stream features (stream network and floodplain)

Stream features (stream network and floodplain) If a variable is added or removed here, then it also has to be added or removed in the subroutine L11_config_set in module mo_restart and in the subroutine set_L11_config in module mo_set_netcdf_restart

Parameters

in	*integer(i4) :: iBasin*	Basin Id

Authors

Luis Samaniego

Date

Dec 2005

stack(nNodes, 2)

Definition at line 1224 of file mo_mrm_net_startup.f90.

References celllength(), movedownonecell(), and moveup().

Referenced by mo_mrm_init::mrm_init().

Here is the call graph for this function:

Here is the caller graph for this function:

15.101.2.13 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 1754 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:

15.101.2.14 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 1581 of file mo_mrm_net_startup.f90.

Referenced by l11_stream_features().

Here is the caller graph for this function:

The documentation for this module was generated from the following file:

- mo_mrm_net_startup.f90

15.102 mo_mrm_objective_function_runoff Module Reference

Objective Functions for Optimization of mHM/mRM against runoff.

Public Member Functions

- real(dp) function, public single_objective_runoff (parameterset, eval, arg1, arg2, arg3)

  Wrapper for objective functions optimizing agains runoff.
- subroutine, public multi_objective_runoff (parameterset, eval, multi_objectives)

  Wrapper for multi-objective functions where at least one is regarding runoff.
- real(dp) function loglikelihood_stddev (parameterset, eval, stddev, stddev_new, likeli_new)

  Logarithmic likelihood function with removed linear trend and Lag(1)-autocorrelation.
- real(dp) function loglikelihood_evin2013_2 (parameterset, eval, regularize)

  Logarithmised likelihood with linear error model and lag(1)-autocorrelation of the relative errors.
- real(dp) function parameter_regularization (paraset, prior, bounds, mask)

  TODO: add description.
- real(dp) function loglikelihood_trend_no_autocorr (parameterset, eval, stddev_old, stddev_new, likeli_new)

  Logarithmic likelihood function with linear trend removed.
- real(dp) function objective_lnnse (parameterset, eval)

  Objective function of logarithmic NSE.
- real(dp) function objective_sse (parameterset, eval)

  Objective function of SSE.
- real(dp) function objective_nse (parameterset, eval)

  Objective function of NSE.
- real(dp) function objective_equal_nse_lnnse (parameterset, eval)

  Objective function equally weighting NSE and lnNSE.
- real(dp) function, dimension(2) multi_objective_nse_lnnse (parameterset, eval)

  Multi-objective function with NSE and lnNSE.
- real(dp) function, dimension(2) multi_objective_lnnse_highflow_lnnse_lowflow (parameterset, eval)

  Multi-objective function with NSE and lnNSE.

- real(dp) function, dimension(2) multi_objective_lnnse_highflow_lnnse_lowflow_2 (parameterset, eval)

  Multi-objective function with NSE and lnNSE.
- real(dp) function, dimension(2) multi_objective_ae_fdc_lsv_nse_djf (parameterset, eval)

  Multi-objective function with absolute error of Flow Duration Curves low-segment volume and nse of DJF's discharge.
- real(dp) function objective_power6_nse_lnnse (parameterset, eval)

  Objective function of combined NSE and lnNSE with power of 5 i.e. the p-norm with p=5.
- real(dp) function objective_kge (parameterset, eval)

  Objective function of KGE.
- real(dp) function objective_multiple_gauges_kge_power6 (parameterset, eval)

  combined objective function based on KGE raised to the power 6
- real(dp) function objective_weighted_nse (parameterset, eval)

  Objective function of weighted NSE.
- subroutine, public extract_runoff (gaugeId, runoff, runoff_agg, runoff_obs, runoff_obs_mask)

  extracts runoff data from global variables

15.102.1 Detailed Description

Objective Functions for Optimization of mHM/mRM against runoff.

This module provides a wrapper for several objective functions used to optimize mRM/mHM against runoff. If the objective contains besides runoff another variable like TWS move it to mHM/mo_objective_function.f90. If it is only regarding runoff implement it here. All the objective functions are supposed to be minimized! (1) SO: Q: 1.0 - NSE (2) SO: Q: 1.0 - lnNSE (3) SO: Q: 1.0 - 0.5∗(NSE+lnNSE) (4) SO: Q: -1.0 ∗ loglikelihood with trend removed from absolute errors and then lag(1)-autocorrelation removed (5) SO: Q: ((1-NSE)∗∗6+(1-lnNSE)∗∗6)∗∗(1/6) (6) SO: Q: SSE (7) SO: Q: -1.0 ∗ loglikelihood with trend removed from absolute errors (8) SO: Q: -1.0 ∗ loglikelihood with trend removed from the relative errors and then lag(1)-autocorrelation removed (9) SO: Q: 1.0 - KGE (Kling-Gupta efficiency measure) (14) SO: Q: sum[((1.0-KGE_i)/ nGauges)∗∗6]∗∗(1/6) > combination of KGE of every gauging station based on a power-6 norm (16) MO: Q: 1st objective: 1.0 - NSE Q: 2nd objective: 1.0 - lnNSE (18) MO: Q: 1st objective: 1.0 - lnNSE(Q_highflow) (95% percentile) Q: 2nd objective: 1.0 - lnNSE(Q_lowflow) (5% of data range) (19) MO: Q: 1st objective: 1.0 - lnNSE(Q_highflow) (non-low flow) Q: 2nd objective: 1.0 - lnNSE(Q_lowflow) (5% of data range)eshold for Q (20) MO: Q: 1st objective: absolute difference in FDC's low-segment volume Q: 2nd objective: 1.0 - NSE of discharge of months DJF (31) SO: Q: 1.0 - wNSE - weighted NSE

Authors

Juliane Mai

Date

Dec 2012

Definition at line 50 of file mo_mrm_objective_function_runoff.f90.

15.102.2 Member Function/Subroutine Documentation

15.102.2.1 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 2053 of file mo_mrm_objective_function_runoff.f90.

References mo_message::message().

Referenced by loglikelihood_evin2013_2(), loglikelihood_stddev(), loglikelihood_trend_no_autocorr(), multi_objective_ae_fdc_lsv_nse_djf(), multi_objective_lnnse_highflow_lnnse_lowflow(), multi_objective_lnnse_highflow_lnnse_lowflow_2(), multi_objective_nse_lnnse(), objective_equal_nse_lnnse(), objective_kge(), mo_objective_function::objective_kge_q_et(), mo_objective_function::objective_kge_q_rmse_et(), mo_objective_function::objective_kge_q_rmse_tws(), mo_objective_function::objective_kge_q_sm_corr(), objective_lnnse(), objective_multiple_gauges_kge_power6(), objective_nse(), objective_power6_nse_lnnse(), objective_sse(), and objective_weighted_nse().

Here is the call graph for this function:

Here is the caller graph for this function:

15.102.2.2 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 429 of file mo_mrm_objective_function_runoff.f90.

References extract_runoff(), and parameter_regularization().

Referenced by single_objective_runoff().

Here is the call graph for this function:

Here is the caller graph for this function:

15.102.2.3 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 287 of file mo_mrm_objective_function_runoff.f90.

References extract_runoff().

Referenced by single_objective_runoff().

Here is the call graph for this function:

Here is the caller graph for this function:

15.102.2.4 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 648 of file mo_mrm_objective_function_runoff.f90.

References extract_runoff().

Referenced by single_objective_runoff().

Here is the call graph for this function:

Here is the caller graph for this function:

15.102.2.5 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 1523 of file mo_mrm_objective_function_runoff.f90.

References mo_julian::dec2date(), extract_runoff(), and mo_mrm_signatures::flowdurationcurve().

Referenced by multi_objective_runoff().

Here is the call graph for this function:

Here is the caller graph for this function:

15.102.2.6 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 1247 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:

15.102.2.7 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 1391 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:

15.102.2.8 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 1142 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:

15.102.2.9 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 213 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(), and multi_objective_nse_lnnse().

Here is the call graph for this function:

15.102.2.10 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 1047 of file mo_mrm_objective_function_runoff.f90.

References extract_runoff().

Referenced by single_objective_runoff().

Here is the call graph for this function:

Here is the caller graph for this function:

15.102.2.11 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 1781 of file mo_mrm_objective_function_runoff.f90.

References extract_runoff().

Referenced by mo_objective_function::objective_kge_q_sm_corr(), and single_objective_runoff().

Here is the call graph for this function:

Here is the caller graph for this function:

15.102.2.12 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 781 of file mo_mrm_objective_function_runoff.f90.

References extract_runoff().

Referenced by single_objective_runoff().

Here is the call graph for this function:

Here is the caller graph for this function:

15.102.2.13 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 1881 of file mo_mrm_objective_function_runoff.f90.

References extract_runoff().

Referenced by single_objective_runoff().

Here is the call graph for this function:

Here is the caller graph for this function:

15.102.2.14 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 956 of file mo_mrm_objective_function_runoff.f90.

References extract_runoff().

Referenced by single_objective_runoff().

Here is the call graph for this function:

Here is the caller graph for this function:

15.102.2.15 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 1683 of file mo_mrm_objective_function_runoff.f90.

References extract_runoff().

Referenced by single_objective_runoff().

Here is the call graph for this function:

Here is the caller graph for this function:

15.102.2.16 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 868 of file mo_mrm_objective_function_runoff.f90.

References extract_runoff().

Referenced by single_objective_runoff().

Here is the call graph for this function:

Here is the caller graph for this function:

15.102.2.17 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 1967 of file mo_mrm_objective_function_runoff.f90.

References extract_runoff().

Referenced by single_objective_runoff().

Here is the call graph for this function:

Here is the caller graph for this function:

15.102.2.18 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 564 of file mo_mrm_objective_function_runoff.f90.

Referenced by loglikelihood_evin2013_2().

Here is the caller graph for this function:

15.102.2.19 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 111 of file mo_mrm_objective_function_runoff.f90.

References loglikelihood_evin2013_2(), loglikelihood_stddev(), loglikelihood_trend_no_autocorr(), mo_message↩ ::message(), objective_equal_nse_lnnse(), objective_kge(), objective_lnnse(), objective_multiple_gauges_kge_↩ power6(), objective_nse(), objective_power6_nse_lnnse(), objective_sse(), and objective_weighted_nse().

Referenced by mhm_driver(), and mrm_driver().

Here is the call graph for this function:

Here is the caller graph for this function:

The documentation for this module was generated from the following file:

- mo_mrm_objective_function_runoff.f90

15.103 mo_mrm_read_config Module Reference

read mRM config

Public Member Functions

- 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.

15.103.1 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

Definition at line 14 of file mo_mrm_read_config.f90.

15.103.2 Member Function/Subroutine Documentation

15.103.2.1 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_nml::close_nml(), mo_message::message(), mo_nml::open_nml(), mo_nml::position_nml(), and read_mrm_routing_params().

Referenced by mo_mrm_init::mrm_init().

Here is the call graph for this function:

Here is the caller graph for this function:

15.103.2.2 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 397 of file mo_mrm_read_config.f90.

References mo_nml::close_nml(), mo_common_functions::in_bound(), mo_message::message(), mo_nml::open↩_nml(), and mo_nml::position_nml().

Referenced by mrm_read_config().

Here is the call graph for this function:

Here is the caller graph for this function:

The documentation for this module was generated from the following file:

- mo_mrm_read_config.f90

15.104 mo_mrm_read_data Module Reference

This module contains all routines to read mRM data from file.

Public Member Functions

- subroutine, public mrm_read_l0_data (do_reinit, do_readlatlon, do_readlcover)

  read L0 data from file
- subroutine, public mrm_read_discharge

  Read discharge timeseries from file.
- subroutine, public mrm_read_total_runoff (iBasin)

read simulated runoff that is to be routed

- subroutine, public mrm_read_bankfull_runoff (iBasin)
- subroutine rotate_fdir_variable (x)

  TODO: add description.

15.104.1 Detailed Description

This module contains all routines to read mRM data from file.

TODO: add description

Authors

Stephan Thober

Date

Aug 2015

Definition at line 14 of file mo_mrm_read_data.f90.

15.104.2 Member Function/Subroutine Documentation

15.104.2.1 subroutine, public mo_mrm_read_data::mrm_read_bankfull_runoff (integer(i4), intent(in) *iBasin*)

Parameters

in	*ibasin*	

Note

The file read in must contain a double precision float variable with the name "Q_bkfl".

Definition at line 432 of file mo_mrm_read_data.f90.

References mo_read_forcing_nc::read_const_forcing_nc().

Referenced by mo_mrm_init::mrm_init().

Here is the call graph for this function:

Here is the caller graph for this function:

15.104.2.2 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 259 of file mo_mrm_read_data.f90.

References mo_message::message(), and mo_read_timeseries::read_timeseries().

Referenced by mo_mrm_init::mrm_init().

Here is the call graph for this function:

Here is the caller graph for this function:

15.104.2.3 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_message::message(), mo_common_read_data::read_dem(), mo_read_latlon::read_latlon(), mo↩_common_read_data::read_lcover(), and rotate_fdir_variable().

Referenced by mo_mrm_init::mrm_init().

Here is the call graph for this function:

Here is the caller graph for this function:

15.104.2.4 subroutine, public mo_mrm_read_data::mrm_read_total_runoff (integer(i4), intent(in) *iBasin*)

read simulated runoff that is to be routed

read spatio-temporal field of total runoff that has been simulated by a hydrologic model or land surface model. This total runoff will then be aggregated to the level 11 resolution and then routed through the stream network.

Parameters

in	*integer(i4) :: iBasin*	basin id

Authors

Stephan Thober

Date

Sep 2015

Definition at line 374 of file mo_mrm_read_data.f90.

References mo_read_forcing_nc::read_forcing_nc().

Referenced by mo_mrm_init::mrm_init().

Here is the call graph for this function:

Here is the caller graph for this function:

15.104.2.5 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 537 of file mo_mrm_read_data.f90.

Referenced by mrm_read_l0_data().

Here is the caller graph for this function:

The documentation for this module was generated from the following file:

- mo_mrm_read_data.f90

15.105 mo_mrm_restart Module Reference

Restart routines.

Public Member Functions

- subroutine, public mrm_write_restart (iBasin, OutPath)

  write routing states and configuration
- subroutine, public mrm_read_restart_states (iBasin, InPath)

  read routing states
- subroutine, public mrm_read_restart_config (iBasin, InPath)

  reads Level 11 configuration from a restart directory

15.105.1 Detailed Description

Restart routines.

This module contains the subroutines for reading and writing routing related variables to file.

Authors

Stephan Thober

Date

Aug 2015

Definition at line 14 of file mo_mrm_restart.f90.

15.105.2 Member Function/Subroutine Documentation

15.105.2.1 subroutine, public mo_mrm_restart::mrm_read_restart_config (integer(i4), intent(in) *iBasin,* character(256), intent(in) *InPath*)

reads Level 11 configuration from a restart directory

read Level 11 configuration variables from a given restart directory and initializes all Level 11 configuration variables, that are initialized in L11_variable_init, contained in module mo_startup.

Parameters

in	*integer(i4) :: iBasin*	number of basin
in	*character(256) :: InPath*	Input Path including trailing slash

Authors

Stephan Thober

Date

Apr 2013

Definition at line 539 of file mo_mrm_restart.f90.

References mo_message::message().

Referenced by mo_mrm_init::mrm_init().

Here is the call graph for this function:

Here is the caller graph for this function:

15.105.2.2 subroutine, public mo_mrm_restart::mrm_read_restart_states (integer(i4), intent(in) *iBasin,* character(256), intent(in) *InPath*)

read routing states

This subroutine reads the routing states from mRM_states_<basin_id>.nc that has to be in the given path directory. This subroutine has to be called directly each time the mHM_eval or mRM_eval is called such that the the states are always the same at the first simulation time step, crucial for optimization.

Parameters

in	*integer(i4) :: iBasin*	number of basin
in	*character(256) :: InPath*	Input Path including trailing slash

Authors

Stephan Thober

Date

Sep 2015

Definition at line 400 of file mo_mrm_restart.f90.

Referenced by mo_mhm_eval::mhm_eval(), and mo_mrm_eval::mrm_eval().

Here is the caller graph for this function:

15.105.2.3 subroutine, public mo_mrm_restart::mrm_write_restart (integer(i4), intent(in) *iBasin,* character(256), dimension(:), intent(in) *OutPath*)

write routing states and configuration

write configuration and state variables to a given restart directory.

Parameters

in	*integer(i4) :: iBasin*	number of basin
in	*character(256), dimension(:) :: OutPath*	list of Output paths per Basin

Authors

Stephan Thober

Date

Aug 2015

Definition at line 54 of file mo_mrm_restart.f90.

References mo_message::message(), 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:

The documentation for this module was generated from the following file:

- mo_mrm_restart.f90

15.106 mo_mrm_river_head Module Reference

Collaboration diagram for mo_mrm_river_head:

Public Member Functions

- subroutine, public init_masked_zeros_l0 (iBasin, data)
- subroutine, public calc_channel_elevation ()
- subroutine, public calc_river_head (iBasin, L11_Qmod, river_head)
- subroutine, public avg_and_write_timestep (iBasin, timestep, data)
- subroutine, public create_output (iBasin, OutPath)

Private Member Functions

- subroutine, private reset_sum (iBasin, data)
- real(dp) function calc_slope (iBasin, elev0, fDir0, i, j)

Private Attributes

- type(ncvariable), dimension(:), allocatable nc_time
- type(ncvariable), dimension(:), allocatable nc_riverhead
- integer(i4), dimension(:), allocatable time_counter
- integer(i4), dimension(:), allocatable sum_counter

15.106.1 Detailed Description

Definition at line 1 of file mo_mrm_river_head.f90.

15.106.2 Member Function/Subroutine Documentation

15.106.2.1 subroutine, public mo_mrm_river_head::avg_and_write_timestep (integer(i4), intent(in) *iBasin,* integer(i4), intent(in) *timestep,* real(dp), dimension(:), intent(inout) *data*)

Definition at line 188 of file mo_mrm_river_head.f90.

References reset_sum().

Referenced by mo_mhm_eval::mhm_eval(), and mo_mrm_eval::mrm_eval().

Here is the call graph for this function:

Here is the caller graph for this function:

15.106.2.2 subroutine, public mo_mrm_river_head::calc_channel_elevation ( )

Definition at line 67 of file mo_mrm_river_head.f90.

References calc_slope().

Referenced by mo_mrm_init::mrm_init().

Here is the call graph for this function:

Here is the caller graph for this function:

15.106.2.3 subroutine, public mo_mrm_river_head::calc_river_head (integer(i4), intent(in) *iBasin,* real(dp), dimension(:), intent(in) *L11_Qmod,* real(dp), dimension(:), intent(inout), allocatable *river_head*)

Definition at line 133 of file mo_mrm_river_head.f90.

Referenced by mo_mhm_eval::mhm_eval(), and mo_mrm_eval::mrm_eval().

Here is the caller graph for this function:

15.106.2.4 real(dp) function mo_mrm_river_head::calc_slope (integer(i4), intent(in) *iBasin,* real(dp), dimension(:,:), intent(in), allocatable *elev0,* integer(i4), dimension(:,:), intent(in), allocatable *fDir0,* integer(i4), intent(in) *i,* integer(i4), intent(in) *j*) [private]

Definition at line 162 of file mo_mrm_river_head.f90.

References mo_mrm_net_startup::celllength(), 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:

15.106.2.5 subroutine, public mo_mrm_river_head::create_output (integer(i4), intent(in) *iBasin,* character(256), intent(in) *OutPath*)

Definition at line 203 of file mo_mrm_river_head.f90.

References mo_julian::dec2date(), mo_grid::geocoordinates(), mo_grid::mapcoordinates(), and mo_message↩ ::message().

Referenced by mo_mrm_init::mrm_init().

Here is the call graph for this function:

Here is the caller graph for this function:

15.106.2.6 subroutine, public mo_mrm_river_head::init_masked_zeros_l0 (integer(i4), intent(in) *iBasin,* real(dp), dimension(:), intent(inout), allocatable *data*)

Definition at line 38 of file mo_mrm_river_head.f90.

References 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:

15.106.2.7 subroutine, private mo_mrm_river_head::reset_sum (integer(i4), intent(in) *iBasin,* real(dp), dimension(:), intent(inout) *data*) [private]

Definition at line 51 of file mo_mrm_river_head.f90.

Referenced by avg_and_write_timestep(), and init_masked_zeros_l0().

Here is the caller graph for this function:

15.106.3 Member Data Documentation

15.106.3.1 type(ncvariable), dimension(:), allocatable mo_mrm_river_head::nc_riverhead [private]

Definition at line 28 of file mo_mrm_river_head.f90.

15.106.3.2 type(ncvariable), dimension(:), allocatable mo_mrm_river_head::nc_time [private]

Definition at line 28 of file mo_mrm_river_head.f90.

15.106.3.3 integer(i4), dimension(:), allocatable mo_mrm_river_head::sum_counter [private]

Definition at line 32 of file mo_mrm_river_head.f90.

15.106.3.4 integer(i4), dimension(:), allocatable mo_mrm_river_head::time_counter [private]

Definition at line 30 of file mo_mrm_river_head.f90.

The documentation for this module was generated from the following file:

- mo_mrm_river_head.f90

15.107 mo_mrm_routing Module Reference

Performs runoff routing for mHM at level L11.

Public Member Functions

- subroutine, public mrm_routing (read_states, processCase, global_routing_param, L1_total_runoff, L1_↩areaCell, L1_L11_Id, L11_areaCell, L11_L1_Id, L11_netPerm, L11_fromN, L11_toN, L11_nOutlets, timestep, tsRoutFactor, nNodes, nInflowGauges, InflowGaugeIndexList, InflowGaugeHeadwater, InflowGaugeNode↩List, InflowDischarge, nGauges, gaugeIndexList, gaugeNodeList, map_flag, L11_length, L11_slope, L11_↩FracFPimp, L11_C1, L11_C2, L11_qOut, L11_qTIN, L11_qTR, L11_qMod, GaugeDischarge)

  route water given runoff
- subroutine l11_runoff_acc (qAll, efecArea, L1_L11_Id, L11_areaCell, L11_L1_Id, TS, map_flag, qAcc)

  total runoff accumulation at L11.
- subroutine add_inflow (nInflowGauges, InflowIndexList, InflowHeadwater, InflowNodeList, QInflow, qOut)

  Adds inflow discharge to the runoff produced at the cell where the inflow is occurring.

- subroutine l11_routing (nNodes, nLinks, netPerm, netLink_fromN, netLink_toN, netLink_C1, netLink_C2, netNode_qOUT, nInflowGauges, InflowHeadwater, InflowNodeList, netNode_qTIN, netNode_qTR, netNode↩_Qmod)

  Performs runoff routing for mHM at L11 upscaled network (Routing Network).

15.107.1 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

Definition at line 16 of file mo_mrm_routing.f90.

15.107.2 Member Function/Subroutine Documentation

15.107.2.1 subroutine mo_mrm_routing::add_inflow (integer(i4), intent(in) *nInflowGauges,* integer(i4), dimension(:), intent(in) *InflowIndexList,* logical, dimension(:), intent(in) *InflowHeadwater,* integer(i4), dimension(:), intent(in) *InflowNodeList,* real(dp), dimension(:), intent(in) *QInflow,* real(dp), dimension(:), intent(inout) *qOut*)

Adds inflow discharge to the runoff produced at the cell where the inflow is occurring.

If a inflow gauge is given, then this routine is adding the values to the runoff produced at the grid cell where the inflow is happening. The values are not directly added to the river network. If this cell is not a headwater then the streamflow produced upstream will be neglected.

Parameters

in	*integer(i4) :: nInflowGauges*	[-] number of inflow points
in	*integer(i4), dimension(:) :: InflowIndexList*	[-] index of inflow points
in	*logical, dimension(:) :: InflowHeadwater*	[-] if to consider headwater cells of inflow gauge
in	*integer(i4), dimension(:) :: InflowNodeList*	[-] L11 ID of inflow points
in	*real(dp), dimension(:) :: QInflow*	[m3 s-1] inflowing water
in,out	*real(dp), dimension(:) :: qOut*	[m3 s-1] Series of attenuated runoff

Authors

Stephan Thober & Matthias Zink

Date

Jul 2016

Definition at line 450 of file mo_mrm_routing.f90.

Referenced by mrm_routing().

Here is the caller graph for this function:

15.107.2.2 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 [**?**]. This simplification of the St. Venant equations is justified in mHM because the potential areas of application of this model would hardly exhibit abruptly changing hydrographs with supercritical flows. The discharge leaving the river reach located on cell i $Q_i^1(t)$ at time step t can be determined by

$$Q_i^1(t) = Q_i^1(t-1) + c_1\left(Q_i^0(t-1) - Q_i^1(t-1)\right) + c_2\left(Q_i^0(t) - Q_i^0(t-1)\right)$$

with

$$Q_i^0(t) = Q_{i'}(t) + Q_{i'}^1(t)$$

$$c_1 = \frac{\Delta t}{\kappa(1-\xi) + \frac{\Delta t}{2}}$$

$$c_2 = \frac{\frac{\Delta t}{2} - \kappa\xi}{\kappa(1-\xi) + \frac{\Delta t}{2}}$$

where Q_i^0 and Q_i^1 denote the discharge entering and leaving the river reach located on cell i respectively. $Q_{i'}$ is the contribution from the upstream cell i'. κ Muskingum travel time parameter. ξ Muskingum attenuation parameter. Δt time interval in hours. t Time index for each Δt interval. To improve performance, a routing sequence "netPerm" is required. This permutation is determined in the mo_init_mrm routine.

TODO: add description

Parameters

in	*integer(i4) :: nNodes*	number of network nodes = nCells1
in	*integer(i4) :: nLinks*	number of stream segment (reaches)
in	*integer(i4), dimension(:) :: netPerm*	routing order of a given basin (permutation)
in	*integer(i4), dimension(:) :: netLink_fromN*	from node
in	*integer(i4), dimension(:) :: netLink_toN*	to node
in	*real(dp), dimension(:) :: netLink_C1*	routing parameter C1 ([**?**] p. 25-41)
in	*real(dp), dimension(:) :: netLink_C2*	routing parameters C2 (id)
in	*real(dp), dimension(:) :: netNode_qOUT*	Total outflow from cells (given basin) L11 at time tt in [m3 s-1]
in	*integer(i4) :: nInflowGauges*	[-] number of inflow points
in	*logical, dimension(:) :: InflowHeadwater*	[-] if to consider headwater cells of inflow gauge
in	*integer(i4), dimension(:) :: InflowNodeList*	[-] L11 ID of inflow points
in,out	*real(dp), dimension(:, :) :: netNode_qTIN*	[m3 s-1] Total inputs at t-1 and t
in,out	*real(dp), dimension(:, :) :: netNode_qTR*	[m3 s-1] Transformed outflow leaving node I (Muskingum)
out	*real(dp), dimension(nNodes) :: netNode_Qmod*	[m3 s-1] Simulated routed discharge

Authors

Luis Samaniego

Date

Dec 2005

Definition at line 567 of file mo_mrm_routing.f90.

Referenced by mrm_routing().

Here is the caller graph for this function:

15.107.2.3 subroutine mo_mrm_routing::l11_runoff_acc (real(dp), dimension(:), intent(in) *qAll,* real(dp), dimension(:), intent(in) *efecArea,* integer(i4), dimension(:), intent(in) *L1_L11_Id,* real(dp), dimension(:), intent(in) *L11_areaCell,* integer(i4), dimension(:), intent(in) *L11_L1_Id,* integer(i4), intent(in) *TS,* logical, intent(in) *map_flag,* real(dp), dimension(:), intent(out) *qAcc*)

total runoff accumulation at L11.

Upscales runoff in space from L1 to L11 if routing resolution is higher than hydrology resolution (map_flag equals .true.) or downscales runoff from L1 to L11 if routing resolution is lower than hydrology resolution.

Parameters

in	*real(dp), dimension(:) :: qall*	total runoff L1 [mm tst-1]
in	*real(dp), dimension(:) :: efecarea*	effective area in [km2] at Level 1
in	*integer(i4), dimension(:) :: L1_L11_Id*	L11 Ids mapped on L1
in	*real(dp), dimension(:) :: L11_areacell*	effective area in [km2] at Level 11
in	*integer(i4), dimension(:) :: L11_L1_Id*	L1 Ids mapped on L11
in	*integer(i4) :: TS*	time step in [s]
in	*logical :: map_flag*	Flag indicating whether routing resolution is higher than hydrologic one
out	*real(dp), dimension(:) :: qAcc*	aggregated runoff at L11 [m3 s-1]

Authors

Luis Samaniego

Date

Jan 2013

Definition at line 343 of file mo_mrm_routing.f90.

Referenced by mrm_routing().

Here is the caller graph for this function:

15.107.2.4 **subroutine, public mo_mrm_routing::mrm_routing (logical, intent(in) *read_states,* integer(i4), intent(in) *processCase,* real(dp), dimension(:), intent(in) *global_routing_param,* real(dp), dimension(:), intent(in) *L1_total_runoff,* real(dp), dimension(:), intent(in) *L1_areaCell,* integer(i4), dimension(:), intent(in) *L1_L11_Id,* real(dp), dimension(:), intent(in) *L11_areaCell,* integer(i4), dimension(:), intent(in) *L11_L1_Id,* integer(i4), dimension(:), intent(in) *L11_netPerm,* integer(i4), dimension(:), intent(in) *L11_fromN,* integer(i4), dimension(:), intent(in) *L11_toN,* integer(i4), intent(in) *L11_nOutlets,* integer(i4), intent(in) *timestep,* real(dp), intent(in) *tsRoutFactor,* integer(i4), intent(in) *nNodes,* integer(i4), intent(in) *nInflowGauges,* integer(i4), dimension(:), intent(in) *InflowGaugeIndexList,* logical, dimension(:), intent(in) *InflowGaugeHeadwater,* integer(i4), dimension(:), intent(in) *InflowGaugeNodeList,* real(dp), dimension(:), intent(in) *InflowDischarge,* integer(i4), intent(in) *nGauges,* integer(i4), dimension(:), intent(in) *gaugeIndexList,* integer(i4), dimension(:), intent(in) *gaugeNodeList,* logical, intent(in) *map_flag,* real(dp), dimension(:), intent(in) *L11_length,* real(dp), dimension(:), intent(in) *L11_slope,* real(dp), dimension(:), intent(in) *L11_FracFPimp,* real(dp), dimension(:), intent(inout) *L11_C1,* real(dp), dimension(:), intent(inout) *L11_C2,* real(dp), dimension(:), intent(inout) *L11_qOut,* real(dp), dimension(:, :), intent(inout) *L11_qTIN,* real(dp), dimension(:, :), intent(inout) *L11_qTR,* real(dp), dimension(:), intent(inout) *L11_qMod,* real(dp), dimension(:), intent(inout) *GaugeDischarge*)**

route water given runoff

This routine first performs mpr for the routing variables if required, then accumulates the runoff to the routing resolution and eventually routes the water in a third step. The last step is repeated multiple times if the routing timestep is smaller than the timestep of the hydrological timestep

Parameters

in	*logical :: read_states*	whether states are derived from restart file
in	*integer(i4) :: processCase*	Process switch for routing
in	*real(dp), dimension(:) :: global_routing↩ _param*	routing parameters
in	*real(dp), dimension(:) :: L1_total_runoff*	total runoff from L1 grid cells

in	*real(dp), dimension(:) :: L1_areaCell*	L1 cell area
in	*integer(i4), dimension(:) :: L1_L11_Id*	L1 cell ids on L11
in	*real(dp), dimension(:) :: L11_areaCell*	L11 cell area
in	*integer(i4), dimension(:) :: L11_L1_Id*	L11 cell ids on L1
in	*integer(i4), dimension(:) :: L11_netPerm*	L11 routing order
in	*integer(i4), dimension(:) :: L11_fromN*	L11 source grid cell order
in	*integer(i4), dimension(:) :: L11_toN*	L11 target grid cell order
in	*integer(i4) :: L11_nOutlets*	L11 number of outlets/sinks
in	*integer(i4) :: timestep*	simulation timestep in [h]
in	*real(dp) :: tsRoutFactor*	factor between routing timestep and hydrological timestep
in	*integer(i4) :: nNodes*	number of nodes
in	*integer(i4) :: nInflowGauges*	number of inflow gauges
in	*integer(i4), dimension(:) :: InflowGauge↩ IndexList*	index list of inflow gauges
in	*logical, dimension(:) :: InflowGauge↩ Headwater*	flag for headwater cell of inflow gauge
in	*integer(i4), dimension(:) :: InflowGauge↩ NodeList*	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 102 of file mo_mrm_routing.f90.

References add_inflow(), l11_routing(), l11_runoff_acc(), mo_runoff::l1_total_runoff(), and mo_mrm_mpr::reg_rout().

Referenced by mo_mhm_eval::mhm_eval(), and mo_mrm_eval::mrm_eval().

Here is the call graph for this function:

Here is the caller graph for this function:

The documentation for this module was generated from the following file:

- mo_mrm_routing.f90

15.108 mo_mrm_signatures Module Reference

Module with calculations for several hydrological signatures.

Public Member Functions

- real(dp) function, dimension(size(lags, 1)), public autocorrelation (data, lags, mask)

  Autocorrelation of a given data series.
- real(dp) function, dimension(size(quantiles, 1)), public flowdurationcurve (data, quantiles, mask, concavity_index, mid_segment_slope, mhigh_segment↩_volume, high_segment_volume, low_segment_volume)

  Flow duration curves.
- subroutine, public limb_densities (data, mask, RLD, DLD)

  Calculates limb densities.
- real(dp) function maximummonthlyflow (data, mask, yr_start, mo_start, dy_start)

  Maximum of average flows per months.
- subroutine, public moments (data, mask, mean_data, stddev_data, median_data, max_data, mean_log, stddev_log, median_log, max_log)

  Moments of data and log-transformed data, e.g. mean and standard deviation.
- real(dp) function, dimension(size(quantiles, 1)), public peakdistribution (data, quantiles, mask, slope_peak_distribution)

  Calculates the peak distribution.
- real(dp) function, public runoffratio (data, basin_area, mask, precip_series, precip_sum, log_data)

  Runoff ratio (accumulated daily discharge [mm/d] / accumulated daily precipitation [mm/d]).
- real(dp) function, public zeroflowratio (data, mask)

  Ratio of zero values to total number of data points.

15.108.1 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

Definition at line 18 of file mo_mrm_signatures.f90.

15.108.2 Member Function/Subroutine Documentation

15.108.2.1 real(dp) function, dimension(size(lags, 1)), public mo_mrm_signatures::autocorrelation (real(dp), dimension(:), intent(in) *data,* integer(i4), dimension(:), intent(in) *lags,* logical, dimension(size(data, 1)), intent(in), optional *mask*)

Autocorrelation of a given data series.

Calculates the autocorrelation of a data series at given lags. An optional argument for masking data points can be given. The function is basically a wrapper of the function autocorr from the module mo_corr. An optional mask of data points can be specified. ADDITIONAL INFORMATION Used as hydrologic signature with lag 1 in Euser, T., Winsemius, H. C., Hrachowitz, M., Fenicia, F., Uhlenbrook, S., & Savenije, H. H. G. (2013). A framework to assess the realism of model structures using hydrological signatures. Hydrology and Earth System Sciences, 17(5), 1893-1912. doi:10.5194/hess-17-1893-2013

Parameters

in	*real(dp), dimension(:) :: data*	Array of data
in	*integer(i4), dimension(:) :: lags*	Array of lags where autocorrelation is requested
in	*logical, dimension(size(data, 1)), optional :: mask*	Mask for data points givenWorks only with 1d double precision input data.

Authors

Juliane Mai

Date

Jun 2015

Definition at line 71 of file mo_mrm_signatures.f90.

15.108.2.2 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:10.1002/2014W↩ R016520

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:

15.108.2.3 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.

References mo_message::message().

Here is the call graph for this function:

15.108.2.4 real(dp) function mo_mrm_signatures::maximummonthlyflow (real(dp), dimension(:), intent(in) *data,* logical, dimension(size(data, 1)), intent(in), optional *mask,* integer(i4), intent(in), optional *yr_start,* integer(i4), intent(in), optional *mo_start,* integer(i4), intent(in), optional *dy_start*)

Maximum of average flows per months.

Maximum of average flow per month is defined as

$$max_{monthlyflow} = Max(F(i), i = 1, ..12)$$

where $f F(i) $ is the average flow of month i. ADDITIONAL INFORMATION used as hydrologic signature in Shafii, M., & Tolson, B. A. (2015). Optimizing hydrological consistency by incorporating hydrological signatures into model calibration objectives. Water Resources Research, 51(5), 3796-3814. doi:10.1002/2014WR016520

Parameters

in	*real(dp), dimension(:) :: data*	array of data
in	*logical, dimension(size(data, 1)), optional :: mask*	mask for data points given
in	*integer(i4), optional :: yr_start*	year of date of first data point given
in	*integer(i4), optional :: mo_start*	month of date of first data point given (default: 1)
in	*integer(i4), optional :: dy_start*	month of date of first data point given (default: 1)

Returns

real(dp) :: MaximumMonthlyFlow — Maximum of average flow per month Works only with 1d double precision input data. Assumes data are daily values.

Authors

Juliane Mai

Date

Jun 2015

Definition at line 513 of file mo_mrm_signatures.f90.

References mo_julian::date2dec(), mo_julian::dec2date(), and mo_message::message().

Here is the call graph for this function:

15.108.2.5 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.

References mo_message::message().

Here is the call graph for this function:

15.108.2.6 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.

15.108.2.7 real(dp) function, public mo_mrm_signatures::runoffratio (real(dp), dimension(:), intent(in) *data,* real(dp), intent(in) *basin_area,* logical, dimension(size(data, 1)), intent(in), optional *mask,* real(dp), dimension(size(data, 1)), intent(in), optional *precip_series,* real(dp), intent(in), optional *precip_sum,* logical, intent(in), optional *log_data*)

Runoff ratio (accumulated daily discharge [mm/d] / accumulated daily precipitation [mm/d]).

The runoff ratio is defined as

$$runoff_ratio = \frac{\sum_{t=1}^{N} q_t}{\sum_{t=1}^{N} p_t}$$

where p_t and q_t are precipitation and discharge, respectively. Therefore, precipitation over the entire basin is required and both discharge and precipitation have to be converted to the same units [mm/d]. Input discharge is given in [m∗∗3/s] as this is mHM default while precipitation has to be given in [mm/km∗∗2 / day]. Either "precip_sum" or "precip_series" has to be specified. If "precip_series" is used the optional mask is also applied to precipitation values. The "precip_sum" is the accumulated "precip_series". Optionally, a mask for the data (=discharge) can be given. If optional "log_data" is set to .true. the runoff ratio will be calculated as

$$runoff_ratio = \frac{\sum_{t=1}^{N} \log(q_t)}{\sum_{t=1}^{N} p_t}$$

where p_t and q_t are precipitation and discharge, respectively. ADDITIONAL INFORMATION

Returns

real(dp), dimension(size(lags,1)) :: RunoffRation — Ratio of discharge and precipitation Used as hydrologic signature in Shafii, M., & Tolson, B. A. (2015). Optimizing hydrological consistency by incorporating hydrological signatures into model calibration objectives. Water Resources Research, 51(5), 3796-3814. doi:10.↩ 1002/2014WR016520

Parameters

in	*real(dp), dimension(:) :: data*	array of data [m∗∗3/s]
in	*real(dp) :: basin_area*	area of basin [km∗∗2]
in	*logical, dimension(size(data, 1)), optional :: mask*	mask for data points given
in	*real(dp), dimension(size(data, 1)), optional :: precip_series*	daily precipitation values [mm/km∗∗2 / day]
in	*real(dp), optional :: precip_sum*	sum of daily precip. values of whole period[mm/km∗∗2 / day]
in	*logical, optional :: log_data*	ratio using logarithmic dataWorks only with 1d double precision input data.

Authors

Juliane Mai

Date

Jun 2015

Definition at line 925 of file mo_mrm_signatures.f90.

References mo_message::message().

Here is the call graph for this function:

15.108.2.8 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.

References mo_message::message().

Here is the call graph for this function:

The documentation for this module was generated from the following file:

- mo_mrm_signatures.f90

15.109 mo_mrm_write Module Reference

write of discharge and restart files

Collaboration diagram for mo_mrm_write:

Public Member Functions

- subroutine, public mrm_write

  write discharge and restart files
- subroutine write_configfile

  This modules writes the results of the configuration into an ASCII-file.
- subroutine write_daily_obs_sim_discharge (Qobs, Qsim)

  Write a file for the daily observed and simulated discharge timeseries during the evaluation period for each gauging station.
- subroutine, public mrm_write_output_fluxes (iBasin, nCells, timeStep_model_outputs, warmingDays, new↩ Time, nTimeSteps, nTStepDay, tt, day, month, year, timestep, mask11, L11_qmod)

  write fluxes to netcdf output files
- subroutine, public mrm_write_optifile (best_OF, best_paramSet, param_names)

  Write briefly final optimization results.
- subroutine, public mrm_write_optinamelist (parameters, maskpara, parameters_name)

  Write final, optimized parameter set in a namelist format.

Private Attributes

- integer(i4) day_counter
- integer(i4) month_counter
- integer(i4) year_counter
- integer(i4) average_counter
- type(outputdataset) nc

15.109.1 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

Definition at line 15 of file mo_mrm_write.f90.

15.109.2 Member Function/Subroutine Documentation

15.109.2.1 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 59 of file mo_mrm_write.f90.

References mo_mrm_restart::mrm_write_restart(), mo_write_ascii::write_configfile(), and write_daily_obs_sim_↩discharge().

Referenced by mhm_driver(), and mrm_driver().

Here is the call graph for this function:

Here is the caller graph for this function:

15.109.2.2 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 787 of file mo_mrm_write.f90.

References mo_message::message().

Referenced by mrm_driver().

Here is the call graph for this function:

Here is the caller graph for this function:

15.109.2.3 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 866 of file mo_mrm_write.f90.

References mo_message::message().

Referenced by mrm_driver().

Here is the call graph for this function:

Here is the caller graph for this function:

15.109.2.4 subroutine, public mo_mrm_write::mrm_write_output_fluxes (integer(i4), intent(in) *iBasin,* integer(i4), intent(in) *nCells,* integer(i4), intent(in) *timeStep_model_outputs,* integer(i4), intent(in) *warmingDays,* real(dp), intent(in) *newTime,* integer(i4), intent(in) *nTimeSteps,* integer(i4), intent(in) *nTStepDay,* integer(i4), intent(in) *tt,* integer(i4), intent(in) *day,* integer(i4), intent(in) *month,* integer(i4), intent(in) *year,* integer(i4), intent(in) *timestep,* logical, dimension(:, :), intent(in), pointer *mask11,* real(dp), dimension(:), intent(in) *L11_qmod*)

write fluxes to netcdf output files

This subroutine creates a netcdf data set for writing L11_QTIN for different time averages.

Parameters

in	*integer(i4) :: iBasin*	
in	*integer(i4) :: nCells*	
in	*integer(i4) :: timeStep_model_outputs*	timestep of model outputs
in	*integer(i4) :: warmingDays*	number of warming days
in	*real(dp) :: newTime*	julian date of next time step
in	*integer(i4) :: nTimeSteps*	number of total timesteps
in	*integer(i4) :: nTStepDay*	number of timesteps per day
in	*integer(i4) :: tt*	current model timestep
in	*integer(i4) :: day*	current day of the year
in	*integer(i4) :: month*	current month of the year
in	*integer(i4) :: year*	current year
in	*integer(i4) :: timestep*	current model time resolution
in	*logical, dimension(:, :) :: mask11*	mask at level 11
in	*real(dp), dimension(:) :: L11_qMod*	current routed streamflow

Authors

Stephan Thober

Date

Aug 2015

Definition at line 636 of file mo_mrm_write.f90.

References mo_julian::caldat().

Referenced by mo_mhm_eval::mhm_eval(), and mo_mrm_eval::mrm_eval().

Here is the call graph for this function:

Here is the caller graph for this function:

15.109.2.5 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_message::message().

Here is the call graph for this function:

15.109.2.6 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 467 of file mo_mrm_write.f90.

References mo_julian::dec2date(), and mo_message::message().

Referenced by mrm_write().

Here is the call graph for this function:

Here is the caller graph for this function:

15.109.3 Member Data Documentation

15.109.3.1 integer(i4) mo_mrm_write::average_counter [private]

Definition at line 33 of file mo_mrm_write.f90.

15.109.3.2 integer(i4) mo_mrm_write::day_counter [private]

Definition at line 30 of file mo_mrm_write.f90.

15.109.3.3 integer(i4) mo_mrm_write::month_counter [private]

Definition at line 31 of file mo_mrm_write.f90.

15.109.3.4 type(outputdataset) mo_mrm_write::nc [private]

Definition at line 34 of file mo_mrm_write.f90.

15.109.3.5 integer(i4) mo_mrm_write::year_counter [private]

Definition at line 32 of file mo_mrm_write.f90.

The documentation for this module was generated from the following file:

- mo_mrm_write.f90

15.110 mo_mrm_write_fluxes_states Module Reference

Creates NetCDF output for different fluxes and state variables of mHM.

Data Types

- interface outputdataset
- interface outputvariable

Public Member Functions

- subroutine updatedataset (self, sidx, eidx, L11_Qmod)

  Update all variables.
- subroutine writetimestep (self, timestep)

  Write all accumulated data.
- type(ncdataset) function createoutputfile (ibasin)

  Create and initialize output file.
- subroutine writevariableattributes (var, long_name, unit)

  Write output variable attributes.

Private Member Functions

- type(outputvariable) function newoutputvariable (nc, name, dtype, dims, ncells, mask, avg)

  Initialize OutputVariable.
- subroutine updatevariable (self, data)

  Update OutputVariable.
- subroutine writevariabletimestep (self, timestep)

  Write timestep to file.
- type(outputdataset) function newoutputdataset (ibasin, mask, nCells)

  Initialize OutputDataset.
- subroutine close (self)

  Close the file.

15.110.1 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

Definition at line 16 of file mo_mrm_write_fluxes_states.f90.

15.110.2 Member Function/Subroutine Documentation

15.110.2.1 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 422 of file mo_mrm_write_fluxes_states.f90.

References mo_message::message().

Here is the call graph for this function:

15.110.2.2 type(ncdataset) function mo_mrm_write_fluxes_states::createoutputfile (integer(i4), intent(in) *ibasin*)

Create and initialize output file.

Create output file, write all non-dynamic variables and global attributes for the given basin.

Returns

type(NcDataset)

Parameters

in	*integer(i4) :: ibasin*	-> basin id

Authors

David Schaefer

Date

June 2015

Definition at line 463 of file mo_mrm_write_fluxes_states.f90.

References mo_julian::dec2date(), mo_grid::geocoordinates(), and mo_grid::mapcoordinates().

Here is the call graph for this function:

15.110.2.3 type(outputdataset) function mo_mrm_write_fluxes_states::newoutputdataset (integer(i4), intent(in) *ibasin,* logical, dimension(:, :), intent(in), pointer *mask,* integer(i4), intent(in) *nCells*) [private]

Initialize OutputDataset.

Create and initialize the output file. If new a new output variable needs to be written, this is the first of two procedures to change (second: updateDataset)

Returns

type(OutputDataset)

Parameters

in	*integer(i4) :: ibasin*	-> basin id
in	*logical, dimension(:, :) :: mask*	
in	*integer(i4) :: nCells*	

Authors

Matthias Zink

Date

Apr 2013

Definition at line 238 of file mo_mrm_write_fluxes_states.f90.

References mo_write_fluxes_states::createoutputfile(), and mo_write_fluxes_states::writevariableattributes().

Here is the call graph for this function:

15.110.2.4 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 basin
in	*logical, dimension(:, :) :: mask*	
in	*logical, optional :: avg*	-> average the data before writing

Authors

David Schaefer

Date

June 2015

Definition at line 99 of file mo_mrm_write_fluxes_states.f90.

15.110.2.5 subroutine mo_mrm_write_fluxes_states::updatedataset (class(outputdataset), intent(inout), target *self*, integer(i4), intent(in) *sidx*, integer(i4), intent(in) *eidx*, real(dp), dimension(:), intent(in) *L11_Qmod*)

Update all variables.

Call the type bound procedure updateVariable for all output variables. If a new output variable needs to be written, this is the second of two procedures to change (first: newOutputDataset)

Parameters

in,out	*class(OutputDataset) :: self*	
in	*integer(i4) :: sidx, eidx*	- start index of the basin related data in L1_∗ arguments
in	*integer(i4) :: sidx, eidx*	- end index of the basin related data in L1_∗ arguments
in	*real(dp), dimension(:) :: L11_Qmod*	

Authors

Matthias Zink

Date

Apr 2013

Definition at line 320 of file mo_mrm_write_fluxes_states.f90.

References mo_write_fluxes_states::outputvariable::updatevariable().

Here is the call graph for this function:

15.110.2.6 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 155 of file mo_mrm_write_fluxes_states.f90.

15.110.2.7 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 378 of file mo_mrm_write_fluxes_states.f90.

15.110.2.8 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 579 of file mo_mrm_write_fluxes_states.f90.

15.110.2.9 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 192 of file mo_mrm_write_fluxes_states.f90.

The documentation for this module was generated from the following file:

- mo_mrm_write_fluxes_states.f90

15.111 mo_ncread Module Reference

Data Types

- interface get_ncvar

Public Member Functions

- integer(i4) function,
  dimension(5), public get_ncdim (Filename, Variable, PrintInfo, ndims)
- subroutine, public get_ncdimatt (Filename, Variable, DimName, DimLen)
- subroutine, public get_ncvaratt (FileName, VarName, AttName, AttValues, fid, dtype)
- integer(i4) function, public ncopen (Fname)
- subroutine, public ncclose (ncid)

Private Member Functions

- subroutine get_ncvar_0d_sp (Filename, VarName, Dat, fid)
- subroutine get_ncvar_0d_dp (Filename, VarName, Dat, fid)
- subroutine get_ncvar_1d_sp (Filename, VarName, Dat, start, a_count, fid)
- subroutine get_ncvar_1d_dp (Filename, VarName, Dat, start, a_count, fid)
- subroutine get_ncvar_2d_sp (Filename, VarName, Dat, start, a_count, fid)
- subroutine get_ncvar_2d_dp (Filename, VarName, Dat, start, a_count, fid)
- subroutine get_ncvar_3d_sp (Filename, VarName, Dat, start, a_count, fid)
- subroutine get_ncvar_3d_dp (Filename, VarName, Dat, start, a_count, fid)
- subroutine get_ncvar_4d_sp (Filename, VarName, Dat, start, a_count, fid)
- subroutine get_ncvar_4d_dp (Filename, VarName, Dat, start, a_count, fid)
- subroutine get_ncvar_5d_sp (Filename, VarName, Dat, start, a_count, fid)
- subroutine get_ncvar_5d_dp (Filename, VarName, Dat, start, a_count, fid)
- subroutine get_ncvar_0d_i4 (Filename, VarName, Dat, fid)
- subroutine get_ncvar_1d_i4 (Filename, VarName, Dat, start, a_count, fid)
- subroutine get_ncvar_2d_i4 (Filename, VarName, Dat, start, a_count, fid)
- subroutine get_ncvar_3d_i4 (Filename, VarName, Dat, start, a_count, fid)
- subroutine get_ncvar_4d_i4 (Filename, VarName, Dat, start, a_count, fid)
- subroutine get_ncvar_5d_i4 (Filename, VarName, Dat, start, a_count, fid)
- subroutine get_ncvar_0d_i1 (Filename, VarName, Dat, fid)
- subroutine get_ncvar_1d_i1 (Filename, VarName, Dat, start, a_count, fid)
- subroutine get_ncvar_2d_i1 (Filename, VarName, Dat, start, a_count, fid)
- subroutine get_ncvar_3d_i1 (Filename, VarName, Dat, start, a_count, fid)
- subroutine get_ncvar_4d_i1 (Filename, VarName, Dat, start, a_count, fid)
- subroutine get_ncvar_5d_i1 (Filename, VarName, Dat, start, a_count, fid)
- subroutine get_info (Varname, ncid, varid, xtype, dl, Info, ndims)
- subroutine check (status)

15.111.1 Detailed Description

Definition at line 1 of file mo_ncread.f90.

15.111.2 Member Function/Subroutine Documentation

15.111.2.1 subroutine mo_ncread::check (integer(i4), intent(in) *status*) [private]

Definition at line 2226 of file mo_ncread.f90.

Referenced by mo_netcdf::close(), mo_ncwrite::close_netcdf(), mo_ncwrite::create_netcdf(), mo_ncwrite::dump↩_netcdf_1d_dp(), mo_ncwrite::dump_netcdf_1d_i4(), mo_ncwrite::dump_netcdf_1d_sp(), mo_ncwrite::dump_↩netcdf_2d_dp(), mo_ncwrite::dump_netcdf_2d_i4(), mo_ncwrite::dump_netcdf_2d_sp(), mo_ncwrite::dump_↩netcdf_3d_dp(), mo_ncwrite::dump_netcdf_3d_i4(), mo_ncwrite::dump_netcdf_3d_sp(), mo_ncwrite::dump_↩netcdf_4d_dp(), mo_ncwrite::dump_netcdf_4d_i4(), mo_ncwrite::dump_netcdf_4d_sp(), mo_ncwrite::dump_↩netcdf_5d_dp(), mo_ncwrite::dump_netcdf_5d_i4(), mo_ncwrite::dump_netcdf_5d_sp(), get_info(), get_ncdim(),

get_ncdimatt(), get_ncvar_0d_dp(), get_ncvar_0d_i1(), get_ncvar_0d_i4(), get_ncvar_0d_sp(), get_ncvar_1d_dp(), get_ncvar_1d_i1(), get_ncvar_1d_i4(), get_ncvar_1d_sp(), get_ncvar_2d_dp(), get_ncvar_2d_i1(), get_ncvar_2d_i4(), get_ncvar_2d_sp(), get_ncvar_3d_dp(), get_ncvar_3d_i1(), get_ncvar_3d_i4(), get_ncvar_3d_sp(), get_ncvar_4d_dp(), get_ncvar_4d_i1(), get_ncvar_4d_i4(), get_ncvar_4d_sp(), get_ncvar_5d_dp(), get_ncvar_5d_i1(), get_ncvar_5d_i4(), get_ncvar_5d_sp(), get_ncvaratt(), mo_netcdf::getdata1df32(), mo_netcdf::getdata1df64(), mo_netcdf::getdata1di16(), mo_netcdf::getdata1di32(), mo_netcdf::getdata1di64(), mo_netcdf::getdata1di8(), mo_netcdf::getdata2df32(), mo_netcdf::getdata2df64(), mo_netcdf::getdata2di16(), mo_netcdf::getdata2di32(), mo_netcdf::getdata2di64(), mo_netcdf::getdata2di8(), mo_netcdf::getdata3df32(), mo_netcdf::getdata3df64(), mo_netcdf::getdata3di16(), mo_netcdf::getdata3di32(), mo_netcdf::getdata3di64(), mo_netcdf::getdata3di8(), mo_netcdf::getdata4df32(), mo_netcdf::getdata4df64(), mo_netcdf::getdata4di16(), mo_netcdf::getdata4di32(), mo_netcdf::getdata4di64(), mo_netcdf::getdata4di8(), mo_netcdf::getdata5df32(), mo_netcdf::getdata5df64(), mo_netcdf::getdata5di16(), mo_netcdf::getdata5di32(), mo_netcdf::getdata5di64(), mo_netcdf::getdata5di8(), mo_netcdf::getdatascalarf32(), mo_netcdf::getdatascalarf64(), mo_netcdf::getdatascalari16(), mo_netcdf::getdatascalari32(), mo_netcdf::getdatascalari64(), mo_netcdf::getdatascalari8(), mo_netcdf::getdimensionbyid(), mo_netcdf::getdimensionbyname(), mo_netcdf::getdimensionlength(), mo_netcdf::getdimensionname(), mo_netcdf::getglobalattributechar(), mo_netcdf::getglobalattributef32(), mo_netcdf::getglobalattributef64(), mo_netcdf::getglobalattributei16(), mo_netcdf::getglobalattributei32(), mo_netcdf::getglobalattributei64(), mo_netcdf::getglobalattributei8(), mo_netcdf::getnodimensions(), mo_netcdf::getnovariables(), mo_netcdf::getunlimiteddimension(), mo_netcdf::getvariableattributechar(), mo_netcdf::getvariableattributef32(), mo_netcdf::getvariableattributef64(), mo_netcdf::getvariableattributei16(), mo_netcdf::getvariableattributei32(), mo_netcdf::getvariableattributei64(), mo_netcdf::getvariableattributei8(), mo_netcdf::getvariablebyname(), mo_netcdf::getvariabledimensions(), mo_netcdf::getvariabledtype(), mo_netcdf::getvariableids(), mo_netcdf::getvariablename(), mo_netcdf::initncdataset(), mo_netcdf::isdatasetunlimited(), ncclose(), ncopen(), mo_ncwrite::open_netcdf(), mo_netcdf::setdata1df32(), mo_netcdf::setdata1df64(), mo_netcdf::setdata1di16(), mo_netcdf::setdata1di32(), mo_netcdf::setdata1di64(), mo_netcdf::setdata1di8(), mo_netcdf::setdata2df32(), mo_netcdf::setdata2df64(), mo_netcdf::setdata2di16(), mo_netcdf::setdata2di32(), mo_netcdf::setdata2di64(), mo_netcdf::setdata2di8(), mo_netcdf::setdata3df32(), mo_netcdf::setdata3df64(), mo_netcdf::setdata3di16(), mo_netcdf::setdata3di32(), mo_netcdf::setdata3di64(), mo_netcdf::setdata3di8(), mo_netcdf::setdata4df32(), mo_netcdf::setdata4df64(), mo_netcdf::setdata4di16(), mo_netcdf::setdata4di32(), mo_netcdf::setdata4di64(), mo_netcdf::setdata4di8(), mo_netcdf::setdata5df32(), mo_netcdf::setdata5df64(), mo_netcdf::setdata5di16(), mo_netcdf::setdata5di32(), mo_netcdf::setdata5di64(), mo_netcdf::setdata5di8(), mo_netcdf::setdatascalarf32(), mo_netcdf::setdatascalarf64(), mo_netcdf::setdatascalari16(), mo_netcdf::setdatascalari32(), mo_netcdf::setdatascalari64(), mo_netcdf::setdatascalari8(), mo_netcdf::setdimension(), mo_netcdf::setglobalattributechar(), mo_netcdf::setglobalattributef32(), mo_netcdf::setglobalattributef64(), mo_netcdf::setglobalattributei16(), mo_netcdf::setglobalattributei32(), mo_netcdf::setglobalattributei64(), mo_netcdf::setglobalattributei8(), mo_netcdf::setvariableattributechar(), mo_netcdf::setvariableattributef32(), mo_netcdf::setvariableattributef64(), mo_netcdf::setvariableattributei16(), mo_netcdf::setvariableattributei32(), mo_netcdf::setvariableattributei64(), mo_netcdf::setvariableattributei8(), mo_netcdf::setvariablewithids(), mo_ncwrite::var2nc_1d_dp(), mo_ncwrite::var2nc_1d_i4(), mo_ncwrite::var2nc_1d_sp(), mo_ncwrite::var2nc_2d_dp(), mo_ncwrite::var2nc_2d_i4(), mo_ncwrite::var2nc_2d_sp(), mo_ncwrite::var2nc_3d_dp(), mo_ncwrite::var2nc_3d_i4(), mo_ncwrite::var2nc_3d_sp(), mo_ncwrite::var2nc_4d_dp(), mo_ncwrite::var2nc_4d_i4(), mo_ncwrite::var2nc_4d_sp(), mo_ncwrite::var2nc_5d_dp(), mo_ncwrite::var2nc_5d_i4(), mo_ncwrite::var2nc_5d_sp(), mo_ncwrite::write_dynamic_netcdf(), and mo_ncwrite::write_static_netcdf().

15.111.2.2 subroutine mo_ncread::get_info (character(len = ∗), intent(in) *Varname,* integer(i4), intent(in) *ncid,* integer(i4), intent(out) *varid,* integer(i4), intent(out) *xtype,* integer(i4), dimension(:), intent(inout), optional *dl,* logical, intent(in), optional *Info,* integer(i4), intent(out), optional *ndims*) [private]

Definition at line 2175 of file mo_ncread.f90.

References check().

Referenced by get_ncdim(), get_ncdimatt(), get_ncvar_0d_dp(), get_ncvar_0d_i1(), get_ncvar_0d_i4(), get_ncvar_0d_sp(), get_ncvar_1d_dp(), get_ncvar_1d_i1(), get_ncvar_1d_i4(), get_ncvar_1d_sp(), get_ncvar_2d_dp(), get_ncvar_2d_i1(), get_ncvar_2d_i4(), get_ncvar_2d_sp(), get_ncvar_3d_dp(), get_ncvar_3d_i1(), get_ncvar_3d_i4(), get_ncvar_3d_sp(), get_ncvar_4d_dp(), get_ncvar_4d_i1(), get_ncvar_4d_i4(), get_ncvar_4d_sp(), get_ncvar_5d_dp(), get_ncvar_5d_i1(), get_ncvar_5d_i4(), and get_ncvar_5d_sp().

Here is the call graph for this function:

Here is the caller graph for this function:

15.111.2.3 integer(i4) function, dimension(5), public mo_ncread::get_ncdim (character(len = ∗), intent(in) *Filename,* character(len = ∗), intent(in) *Variable,* logical, intent(in), optional *PrintInfo,* integer(i4), intent(out), optional *ndims*)

Definition at line 153 of file mo_ncread.f90.

References check(), and get_info().

Referenced by get_ncdimatt(), get_ncvar_1d_dp(), get_ncvar_1d_i1(), get_ncvar_1d_i4(), get_ncvar_1d_sp(), get↩

_ncvar_2d_dp(), get_ncvar_2d_i1(), get_ncvar_2d_i4(), get_ncvar_2d_sp(), get_ncvar_3d_dp(), get_ncvar_3d_i1(), get_ncvar_3d_i4(), get_ncvar_3d_sp(), get_ncvar_4d_dp(), get_ncvar_4d_i1(), get_ncvar_4d_i4(), get_ncvar_4d↩_sp(), get_ncvar_5d_dp(), get_ncvar_5d_i1(), get_ncvar_5d_i4(), and get_ncvar_5d_sp().

Here is the call graph for this function:

Here is the caller graph for this function:

15.111.2.4 subroutine, public mo_ncread::get_ncdimatt (character(len = ∗), intent(in) *Filename,* character(len = ∗), intent(in) *Variable,* character(len = ∗), dimension(:), intent(out), allocatable *DimName,* integer(i4), dimension(:), intent(out), optional, allocatable *DimLen*)

Definition at line 235 of file mo_ncread.f90.

References check(), get_info(), and get_ncdim().

Here is the call graph for this function:

15.111.2.5 subroutine mo_ncread::get_ncvar_0d_dp (character(len = ∗), intent(in) *Filename,* character(len = ∗), intent(in) *VarName,* real(dp), intent(inout) *Dat,* integer(i4), intent(in), optional *fid*) [private]

Definition at line 437 of file mo_ncread.f90.

References check(), and get_info().

Here is the call graph for this function:

15.111.2.6 subroutine mo_ncread::get_ncvar_0d_i1 (character(len = ∗), intent(in) *Filename,* character(len = ∗), intent(in) *VarName,* integer(1), intent(inout) *Dat,* integer(i4), intent(in), optional *fid*) [private]

Definition at line 1678 of file mo_ncread.f90.

References check(), and get_info().

Here is the call graph for this function:

15.111.2.7 subroutine mo_ncread::get_ncvar_0d_i4 (character(len = ∗), intent(in) *Filename,* character(len = ∗), intent(in) *VarName,* integer(i4), intent(inout) *Dat,* integer(i4), intent(in), optional *fid*) [private]

Definition at line 1259 of file mo_ncread.f90.

References check(), and get_info().

Here is the call graph for this function:

15.111.2.8 subroutine mo_ncread::get_ncvar_0d_sp (character(len = ∗), intent(in) *Filename,* character(len = ∗), intent(in) *VarName,* real(sp), intent(inout) *Dat,* integer(i4), intent(in), optional *fid*) `[private]`

Definition at line 398 of file mo_ncread.f90.

References check(), and get_info().

Here is the call graph for this function:

15.111.2.9 subroutine mo_ncread::get_ncvar_1d_dp (character(len = ∗), intent(in) *Filename,* character(len = ∗), intent(in) *VarName,* real(dp), dimension(:), intent(inout), allocatable *Dat,* integer(i4), dimension(:), intent(in), optional *start,* integer(i4), dimension(:), intent(in), optional *a_count,* integer(i4), intent(in), optional *fid*) `[private]`

Definition at line 554 of file mo_ncread.f90.

References check(), get_info(), and get_ncdim().

Here is the call graph for this function:

15.111.2.10 subroutine mo_ncread::get_ncvar_1d_i1 (character(len = ∗), intent(in) *Filename,* character(len = ∗), intent(in) *VarName,* integer(1), dimension(:), intent(inout), allocatable *Dat,* integer(i4), dimension(:), intent(in), optional *start,* integer(i4), dimension(:), intent(in), optional *a_count,* integer(i4), intent(in), optional *fid*) `[private]`

Definition at line 1718 of file mo_ncread.f90.

References check(), get_info(), and get_ncdim().

Here is the call graph for this function:

15.111.2.11 subroutine mo_ncread::get_ncvar_1d_i4 (character(len = ∗), intent(in) *Filename,* character(len = ∗), intent(in) *VarName,* integer(i4), dimension(:), intent(inout), allocatable *Dat,* integer(i4), dimension(:), intent(in), optional *start,* integer(i4), dimension(:), intent(in), optional *a_count,* integer(i4), intent(in), optional *fid*) [private]

Definition at line 1299 of file mo_ncread.f90.

References check(), get_info(), and get_ncdim().

Here is the call graph for this function:

15.111.2.12 subroutine mo_ncread::get_ncvar_1d_sp (character(len = ∗), intent(in) *Filename,* character(len = ∗), intent(in) *VarName,* real(sp), dimension(:), intent(inout), allocatable *Dat,* integer(i4), dimension(:), intent(in), optional *start,* integer(i4), dimension(:), intent(in), optional *a_count,* integer(i4), intent(in), optional *fid*) [private]

Definition at line 476 of file mo_ncread.f90.

References check(), get_info(), and get_ncdim().

Here is the call graph for this function:

15.111.2.13 subroutine mo_ncread::get_ncvar_2d_dp (character(len = ∗), intent(in) *Filename,* character(len = ∗), intent(in) *VarName,* real(dp), dimension(:, :), intent(inout), allocatable *Dat,* integer(i4), dimension(:), intent(in), optional *start,* integer(i4), dimension(:), intent(in), optional *a_count,* integer(i4), intent(in), optional *fid*) [private]

Definition at line 711 of file mo_ncread.f90.

References check(), get_info(), and get_ncdim().

Here is the call graph for this function:

15.111.2.14 subroutine mo_ncread::get_ncvar_2d_i1 (character(len = ∗), intent(in) *Filename,* character(len = ∗), intent(in) *VarName,* integer(1), dimension(:, :), intent(inout), allocatable *Dat,* integer(i4), dimension(:), intent(in), optional *start,* integer(i4), dimension(:), intent(in), optional *a_count,* integer(i4), intent(in), optional *fid*) [private]

Definition at line 1794 of file mo_ncread.f90.

References check(), get_info(), and get_ncdim().

Here is the call graph for this function:

15.111.2.15 subroutine mo_ncread::get_ncvar_2d_i4 (character(len = ∗), intent(in) *Filename,* character(len = ∗), intent(in) *VarName,* integer(i4), dimension(:, :), intent(inout), allocatable *Dat,* integer(i4), dimension(:), intent(in), optional *start,* integer(i4), dimension(:), intent(in), optional *a_count,* integer(i4), intent(in), optional *fid*) [private]

Definition at line 1375 of file mo_ncread.f90.

References check(), get_info(), and get_ncdim().

Here is the call graph for this function:

15.111.2.16 subroutine mo_ncread::get_ncvar_2d_sp (character(len = ∗), intent(in) *Filename,* character(len = ∗), intent(in) *VarName,* real(sp), dimension(:, :), intent(inout), allocatable *Dat,* integer(i4), dimension(:), intent(in), optional *start,* integer(i4), dimension(:), intent(in), optional *a_count,* integer(i4), intent(in), optional *fid*) [private]

Definition at line 632 of file mo_ncread.f90.

References check(), get_info(), and get_ncdim().

Here is the call graph for this function:

15.111.2.17 subroutine mo_ncread::get_ncvar_3d_dp (character(len = ∗), intent(in) *Filename,* character(len = ∗), intent(in) *VarName,* real(dp), dimension(:, :, :), intent(inout), allocatable *Dat,* integer(i4), dimension(:), intent(in), optional *start,* integer(i4), dimension(:), intent(in), optional *a_count,* integer(i4), intent(in), optional *fid*) [private]

Definition at line 869 of file mo_ncread.f90.

References check(), get_info(), and get_ncdim().

Here is the call graph for this function:

15.111.2.18 subroutine mo_ncread::get_ncvar_3d_i1 (character(len = ∗), intent(in) *Filename,* character(len = ∗), intent(in) *VarName,* integer(1), dimension(:, :, :), intent(inout), allocatable *Dat,* integer(i4), dimension(:), intent(in), optional *start,* integer(i4), dimension(:), intent(in), optional *a_count,* integer(i4), intent(in), optional *fid*) [private]

Definition at line 1870 of file mo_ncread.f90.

References check(), get_info(), and get_ncdim().

Here is the call graph for this function:

15.111.2.19 subroutine mo_ncread::get_ncvar_3d_i4 (character(len = ∗), intent(in) *Filename,* character(len = ∗), intent(in) *VarName,* integer(i4), dimension(:, :, :), intent(inout), allocatable *Dat,* integer(i4), dimension(:), intent(in), optional *start,* integer(i4), dimension(:), intent(in), optional *a_count,* integer(i4), intent(in), optional *fid*) [private]

Definition at line 1451 of file mo_ncread.f90.

References check(), get_info(), and get_ncdim().

Here is the call graph for this function:

15.111.2.20 subroutine mo_ncread::get_ncvar_3d_sp (character(len = ∗), intent(in) *Filename,* character(len = ∗), intent(in) *VarName,* real(sp), dimension(:, :, :), intent(inout), allocatable *Dat,* integer(i4), dimension(:), intent(in), optional *start,* integer(i4), dimension(:), intent(in), optional *a_count,* integer(i4), intent(in), optional *fid*) [private]

Definition at line 789 of file mo_ncread.f90.

References check(), get_info(), and get_ncdim().

Here is the call graph for this function:

15.111.2.21 subroutine mo_ncread::get_ncvar_4d_dp (character(len = ∗), intent(in) *Filename,* character(len = ∗), intent(in) *VarName,* real(dp), dimension(:, :, :, :), intent(inout), allocatable *Dat,* integer(i4), dimension(:), intent(in), optional *start,* integer(i4), dimension(:), intent(in), optional *a_count,* integer(i4), intent(in), optional *fid*) [private]

Definition at line 1025 of file mo_ncread.f90.

References check(), get_info(), and get_ncdim().

Here is the call graph for this function:

15.111.2.22 subroutine mo_ncread::get_ncvar_4d_i1 (character(len = ∗), intent(in) *Filename,* character(len = ∗), intent(in) *VarName,* integer(1), dimension(:, :, :, :), intent(inout), allocatable *Dat,* integer(i4), dimension(:), intent(in), optional *start,* integer(i4), dimension(:), intent(in), optional *a_count,* integer(i4), intent(in), optional *fid*) [private]

Definition at line 1946 of file mo_ncread.f90.

References check(), get_info(), and get_ncdim().

Here is the call graph for this function:

15.111.2.23 subroutine mo_ncread::get_ncvar_4d_i4 (character(len = ∗), intent(in) *Filename,* character(len = ∗), intent(in) *VarName,* integer(i4), dimension(:, :, :, :), intent(inout), allocatable *Dat,* integer(i4), dimension(:), intent(in), optional *start,* integer(i4), dimension(:), intent(in), optional *a_count,* integer(i4), intent(in), optional *fid*) [private]

Definition at line 1527 of file mo_ncread.f90.

References check(), get_info(), and get_ncdim().

Here is the call graph for this function:

15.111.2.24 subroutine mo_ncread::get_ncvar_4d_sp (character(len = ∗), intent(in) *Filename,* character(len = ∗), intent(in) *VarName,* real(sp), dimension(:, :, :, :), intent(inout), allocatable *Dat,* integer(i4), dimension(:), intent(in), optional *start,* integer(i4), dimension(:), intent(in), optional *a_count,* integer(i4), intent(in), optional *fid*) [private]

Definition at line 947 of file mo_ncread.f90.

References check(), get_info(), and get_ncdim().

Here is the call graph for this function:

15.111.2.25 subroutine mo_ncread::get_ncvar_5d_dp (character(len = ∗), intent(in) *Filename,* character(len = ∗), intent(in) *VarName,* real(dp), dimension(:, :, :, :, :), intent(inout), allocatable *Dat,* integer(i4), dimension(:), intent(in), optional *start,* integer(i4), dimension(:), intent(in), optional *a_count,* integer(i4), intent(in), optional *fid*) [private]

Definition at line 1181 of file mo_ncread.f90.

References check(), get_info(), and get_ncdim().

Here is the call graph for this function:

15.111.2.26 subroutine mo_ncread::get_ncvar_5d_i1 (character(len = ∗), intent(in) *Filename,* character(len = ∗), intent(in) *VarName,* integer(1), dimension(:, :, :, :), intent(inout), allocatable *Dat,* integer(i4), dimension(:), intent(in), optional *start,* integer(i4), dimension(:), intent(in), optional *a_count,* integer(i4), intent(in), optional *fid*) [private]

Definition at line 2022 of file mo_ncread.f90.

References check(), get_info(), and get_ncdim().

Here is the call graph for this function:

15.111.2.27 subroutine mo_ncread::get_ncvar_5d_i4 (character(len = ∗), intent(in) *Filename,* character(len = ∗), intent(in) *VarName,* integer(i4), dimension(:, :, :, :), intent(inout), allocatable *Dat,* integer(i4), dimension(:), intent(in), optional *start,* integer(i4), dimension(:), intent(in), optional *a_count,* integer(i4), intent(in), optional *fid*) [private]

Definition at line 1603 of file mo_ncread.f90.

References check(), get_info(), and get_ncdim().

Here is the call graph for this function:

15.111.2.28 subroutine mo_ncread::get_ncvar_5d_sp (character(len = ∗), intent(in) *Filename,* character(len = ∗), intent(in) *VarName,* real(sp), dimension(:, :, :, :, :), intent(inout), allocatable *Dat,* integer(i4), dimension(:), intent(in), optional *start,* integer(i4), dimension(:), intent(in), optional *a_count,* integer(i4), intent(in), optional *fid*) `[private]`

Definition at line 1103 of file mo_ncread.f90.

References check(), get_info(), and get_ncdim().

Here is the call graph for this function:

15.111.2.29 subroutine, public mo_ncread::get_ncvaratt (character(len = ∗), intent(in) *FileName,* character(len = ∗), intent(in) *VarName,* character(len = ∗), intent(in) *AttName,* character(len = ∗), intent(out) *AttValues,* integer(i4), intent(in), optional *fid,* integer(i4), intent(out), optional *dtype*)

Definition at line 326 of file mo_ncread.f90.

References check().

Here is the call graph for this function:

15.111.2.30 subroutine, public mo_ncread::ncclose (integer(i4), intent(in) *ncid*)

Definition at line 2150 of file mo_ncread.f90.

References check().

Here is the call graph for this function:

15.111.2.31 integer(i4) function, public mo_ncread::ncopen (character(len = ∗), intent(in) *Fname*)

Definition at line 2119 of file mo_ncread.f90.

References check().

Here is the call graph for this function:

The documentation for this module was generated from the following file:

- mo_ncread.f90

15.112 mo_ncwrite Module Reference

Collaboration diagram for mo_ncwrite:

Data Types

- type attribute
- type dims
- interface dump_netcdf

- interface var2nc

  Extended dump_netcdf for multiple variables.
- type variable

Public Member Functions

- subroutine, public close_netcdf (ncId)
- subroutine, public create_netcdf (Filename, ncid, lfs, netcdf4, deflate_level)
- subroutine, public write_dynamic_netcdf (ncId, irec)
- subroutine, public write_static_netcdf (ncId)

Public Attributes

- integer(i4), parameter, public nmaxdim = 5
- integer(i4), parameter, public nmaxatt = 20
- integer(i4), parameter, public maxlen = 256
- integer(i4), parameter, public ngatt = 20
- integer(i4), parameter, public nattdim = 2
- integer(i4), public nvars
- integer(i4), public ndims
- type(dims), dimension(:),
  allocatable, public dnc
- type(variable), dimension(:),
  allocatable, public v
- type(attribute), dimension(ngatt),
  public gatt

Private Member Functions

- subroutine dump_netcdf_1d_sp (filename, arr, append, lfs, netcdf4, deflate_level)
- subroutine dump_netcdf_2d_sp (filename, arr, append, lfs, netcdf4, deflate_level)
- subroutine dump_netcdf_3d_sp (filename, arr, append, lfs, netcdf4, deflate_level)
- subroutine dump_netcdf_4d_sp (filename, arr, append, lfs, netcdf4, deflate_level)
- subroutine dump_netcdf_5d_sp (filename, arr, append, lfs, netcdf4, deflate_level)
- subroutine dump_netcdf_1d_dp (filename, arr, append, lfs, netcdf4, deflate_level)
- subroutine dump_netcdf_2d_dp (filename, arr, append, lfs, netcdf4, deflate_level)
- subroutine dump_netcdf_3d_dp (filename, arr, append, lfs, netcdf4, deflate_level)
- subroutine dump_netcdf_4d_dp (filename, arr, append, lfs, netcdf4, deflate_level)
- subroutine dump_netcdf_5d_dp (filename, arr, append, lfs, netcdf4, deflate_level)
- subroutine dump_netcdf_1d_i4 (filename, arr, append, lfs, netcdf4, deflate_level)
- subroutine dump_netcdf_2d_i4 (filename, arr, append, lfs, netcdf4, deflate_level)
- subroutine dump_netcdf_3d_i4 (filename, arr, append, lfs, netcdf4, deflate_level)
- subroutine dump_netcdf_4d_i4 (filename, arr, append, lfs, netcdf4, deflate_level)
- subroutine dump_netcdf_5d_i4 (filename, arr, append, lfs, netcdf4, deflate_level)
- subroutine var2nc_1d_i4 (f_name, arr, dnames, v_name, dim_unlimited, long_name, units, missing_value, attributes, create, ncid, nrec)
- subroutine var2nc_1d_sp (f_name, arr, dnames, v_name, dim_unlimited, long_name, units, missing_value, attributes, create, ncid, nrec)
- subroutine var2nc_1d_dp (f_name, arr, dnames, v_name, dim_unlimited, long_name, units, missing_value, attributes, create, ncid, nrec)
- subroutine var2nc_2d_i4 (f_name, arr, dnames, v_name, dim_unlimited, long_name, units, missing_value, attributes, create, ncid, nrec)
- subroutine var2nc_2d_sp (f_name, arr, dnames, v_name, dim_unlimited, long_name, units, missing_value, attributes, create, ncid, nrec)

- subroutine var2nc_2d_dp (f_name, arr, dnames, v_name, dim_unlimited, long_name, units, missing_value, attributes, create, ncid, nrec)
- subroutine var2nc_3d_i4 (f_name, arr, dnames, v_name, dim_unlimited, long_name, units, missing_value, attributes, create, ncid, nrec)
- subroutine var2nc_3d_sp (f_name, arr, dnames, v_name, dim_unlimited, long_name, units, missing_value, attributes, create, ncid, nrec)
- subroutine var2nc_3d_dp (f_name, arr, dnames, v_name, dim_unlimited, long_name, units, missing_value, attributes, create, ncid, nrec)
- subroutine var2nc_4d_i4 (f_name, arr, dnames, v_name, dim_unlimited, long_name, units, missing_value, attributes, create, ncid, nrec)
- subroutine var2nc_4d_sp (f_name, arr, dnames, v_name, dim_unlimited, long_name, units, missing_value, attributes, create, ncid, nrec)
- subroutine var2nc_4d_dp (f_name, arr, dnames, v_name, dim_unlimited, long_name, units, missing_value, attributes, create, ncid, nrec)
- subroutine var2nc_5d_i4 (f_name, arr, dnames, v_name, dim_unlimited, long_name, units, missing_value, attributes, create, ncid, nrec)
- subroutine var2nc_5d_sp (f_name, arr, dnames, v_name, dim_unlimited, long_name, units, missing_value, attributes, create, ncid, nrec)
- subroutine var2nc_5d_dp (f_name, arr, dnames, v_name, dim_unlimited, long_name, units, missing_value, attributes, create, ncid, nrec)
- integer(i4) function open_netcdf (f_name, create)
- subroutine check (status)

15.112.1 Detailed Description

Definition at line 1 of file mo_ncwrite.f90.

15.112.2 Member Function/Subroutine Documentation

15.112.2.1 subroutine mo_ncwrite::check (integer(i4), intent(in) *status*) [private]

Definition at line 5660 of file mo_ncwrite.f90.

15.112.2.2 subroutine, public mo_ncwrite::close_netcdf (integer(i4), intent(in) *ncid*)

Definition at line 346 of file mo_ncwrite.f90.

References mo_ncread::check().

Referenced by var2nc_1d_dp(), var2nc_1d_i4(), var2nc_1d_sp(), var2nc_2d_dp(), var2nc_2d_i4(), var2nc_2d_sp(), var2nc_3d_dp(), var2nc_3d_i4(), var2nc_3d_sp(), var2nc_4d_dp(), var2nc_4d_i4(), var2nc_4d_sp(), var2nc_5d_↩dp(), var2nc_5d_i4(), and var2nc_5d_sp().

Here is the call graph for this function:

Here is the caller graph for this function:

15.112.2.3 subroutine, public mo_ncwrite::create_netcdf (character(len = ∗), intent(in) *Filename,* integer(i4), intent(out) *ncid,* logical, intent(in), optional *lfs,* logical, intent(in), optional *netcdf4,* integer(i4), intent(in), optional *deflate_level*)

Definition at line 401 of file mo_ncwrite.f90.

References mo_ncread::check(), and mo_string_utils::nonull().

Here is the call graph for this function:

15.112.2.4 subroutine mo_ncwrite::dump_netcdf_1d_dp (character(len = ∗), intent(in) *filename,* real(dp), dimension(:), intent(in) *arr,* logical, intent(in), optional *append,* logical, intent(in), optional *lfs,* logical, intent(in), optional *netcdf4,* integer(i4), intent(in), optional *deflate_level*) `[private]`

Definition at line 1261 of file mo_ncwrite.f90.

References mo_ncread::check().

Here is the call graph for this function:

15.112.2.5 subroutine mo_ncwrite::dump_netcdf_1d_i4 (character(len = ∗), intent(in) *filename,* integer(i4), dimension(:), intent(in) *arr,* logical, intent(in), optional *append,* logical, intent(in), optional *lfs,* logical, intent(in), optional *netcdf4,* integer(i4), intent(in), optional *deflate_level*) `[private]`

Definition at line 2009 of file mo_ncwrite.f90.

References mo_ncread::check().

Here is the call graph for this function:

15.112.2.6 subroutine mo_ncwrite::dump_netcdf_1d_sp (character(len = ∗), intent(in) *filename,* real(sp), dimension(:), intent(in) *arr,* logical, intent(in), optional *append,* logical, intent(in), optional *lfs,* logical, intent(in), optional *netcdf4,* integer(i4), intent(in), optional *deflate_level*) [private]

Definition at line 514 of file mo_ncwrite.f90.

References mo_ncread::check().

Here is the call graph for this function:

15.112.2.7 subroutine mo_ncwrite::dump_netcdf_2d_dp (character(len = ∗), intent(in) *filename,* real(dp), dimension(:, :), intent(in) *arr,* logical, intent(in), optional *append,* logical, intent(in), optional *lfs,* logical, intent(in), optional *netcdf4,* integer(i4), intent(in), optional *deflate_level*) [private]

Definition at line 1411 of file mo_ncwrite.f90.

References mo_ncread::check().

Here is the call graph for this function:

15.112.2.8 subroutine mo_ncwrite::dump_netcdf_2d_i4 (character(len = ∗), intent(in) *filename,* integer(i4), dimension(:, :), intent(in) *arr,* logical, intent(in), optional *append,* logical, intent(in), optional *lfs,* logical, intent(in), optional *netcdf4,* integer(i4), intent(in), optional *deflate_level*) [private]

Definition at line 2159 of file mo_ncwrite.f90.

References mo_ncread::check().

Here is the call graph for this function:

15.112.2.9 subroutine mo_ncwrite::dump_netcdf_2d_sp (character(len = ∗), intent(in) *filename,* real(sp), dimension(:, :), intent(in) *arr,* logical, intent(in), optional *append,* logical, intent(in), optional *lfs,* logical, intent(in), optional *netcdf4,* integer(i4), intent(in), optional *deflate_level*) `[private]`

Definition at line 664 of file mo_ncwrite.f90.

References mo_ncread::check().

Here is the call graph for this function:

15.112.2.10 subroutine mo_ncwrite::dump_netcdf_3d_dp (character(len = ∗), intent(in) *filename,* real(dp), dimension(:, :, :), intent(in) *arr,* logical, intent(in), optional *append,* logical, intent(in), optional *lfs,* logical, intent(in), optional *netcdf4,* integer(i4), intent(in), optional *deflate_level*) `[private]`

Definition at line 1561 of file mo_ncwrite.f90.

References mo_ncread::check().

Here is the call graph for this function:

15.112.2.11 subroutine mo_ncwrite::dump_netcdf_3d_i4 (character(len = *), intent(in) *filename,* integer(i4), dimension(:, :, :), intent(in) *arr,* logical, intent(in), optional *append,* logical, intent(in), optional *lfs,* logical, intent(in), optional *netcdf4,* integer(i4), intent(in), optional *deflate_level*) `[private]`

Definition at line 2309 of file mo_ncwrite.f90.

References mo_ncread::check().

Here is the call graph for this function:

15.112.2.12 subroutine mo_ncwrite::dump_netcdf_3d_sp (character(len = *), intent(in) *filename,* real(sp), dimension(:, :, :), intent(in) *arr,* logical, intent(in), optional *append,* logical, intent(in), optional *lfs,* logical, intent(in), optional *netcdf4,* integer(i4), intent(in), optional *deflate_level*) `[private]`

Definition at line 814 of file mo_ncwrite.f90.

References mo_ncread::check().

Here is the call graph for this function:

15.112.2.13 subroutine mo_ncwrite::dump_netcdf_4d_dp (character(len = *), intent(in) *filename,* real(dp), dimension(:, :, :, :), intent(in) *arr,* logical, intent(in), optional *append,* logical, intent(in), optional *lfs,* logical, intent(in), optional *netcdf4,* integer(i4), intent(in), optional *deflate_level*) `[private]`

Definition at line 1711 of file mo_ncwrite.f90.

References mo_ncread::check().

Here is the call graph for this function:

15.112.2.14 **subroutine mo_ncwrite::dump_netcdf_4d_i4 (character(len = ∗), intent(in) *filename,* integer(i4), dimension(:, :, :, :), intent(in) *arr,* logical, intent(in), optional *append,* logical, intent(in), optional *lfs,* logical, intent(in), optional *netcdf4,* integer(i4), intent(in), optional *deflate_level*)** `[private]`

Definition at line 2458 of file mo_ncwrite.f90.

References mo_ncread::check().

Here is the call graph for this function:

15.112.2.15 **subroutine mo_ncwrite::dump_netcdf_4d_sp (character(len = ∗), intent(in) *filename,* real(sp), dimension(:, :, :, :), intent(in) *arr,* logical, intent(in), optional *append,* logical, intent(in), optional *lfs,* logical, intent(in), optional *netcdf4,* integer(i4), intent(in), optional *deflate_level*)** `[private]`

Definition at line 963 of file mo_ncwrite.f90.

References mo_ncread::check().

Here is the call graph for this function:

15.112.2.16 **subroutine mo_ncwrite::dump_netcdf_5d_dp (character(len = ∗), intent(in) *filename,* real(dp), dimension(:, :, :, :, :), intent(in) *arr,* logical, intent(in), optional *append,* logical, intent(in), optional *lfs,* logical, intent(in), optional *netcdf4,* integer(i4), intent(in), optional *deflate_level*)** `[private]`

Definition at line 1860 of file mo_ncwrite.f90.

References mo_ncread::check().

Here is the call graph for this function:

15.112.2.17 **subroutine mo_ncwrite::dump_netcdf_5d_i4 (character(len = ∗), intent(in) *filename,* integer(i4), dimension(:, :, :, :, :), intent(in) *arr,* logical, intent(in), optional *append,* logical, intent(in), optional *lfs,* logical, intent(in), optional *netcdf4,* integer(i4), intent(in), optional *deflate_level*)** `[private]`

Definition at line 2607 of file mo_ncwrite.f90.

References mo_ncread::check().

Here is the call graph for this function:

15.112.2.18 **subroutine mo_ncwrite::dump_netcdf_5d_sp (character(len = ∗), intent(in) *filename,* real(sp), dimension(:, :, :, :, :), intent(in) *arr,* logical, intent(in), optional *append,* logical, intent(in), optional *lfs,* logical, intent(in), optional *netcdf4,* integer(i4), intent(in), optional *deflate_level*)** `[private]`

Definition at line 1112 of file mo_ncwrite.f90.

References mo_ncread::check().

Here is the call graph for this function:

15.112.2.19 integer(i4) function mo_ncwrite::open_netcdf (character(len = ∗), intent(in) *f_name,* logical, intent(in) *create*) [private]

Definition at line 5639 of file mo_ncwrite.f90.

References mo_ncread::check().

Referenced by var2nc_1d_dp(), var2nc_1d_i4(), var2nc_1d_sp(), var2nc_2d_dp(), var2nc_2d_i4(), var2nc_2d_sp(), var2nc_3d_dp(), var2nc_3d_i4(), var2nc_3d_sp(), var2nc_4d_dp(), var2nc_4d_i4(), var2nc_4d_sp(), var2nc_5d_↩dp(), var2nc_5d_i4(), and var2nc_5d_sp().

Here is the call graph for this function:

Here is the caller graph for this function:

15.112.2.20 subroutine mo_ncwrite::var2nc_1d_dp (character(len = ∗), intent(in) *f_name,* real(dp), dimension(:), intent(in) *arr,* character(len = ∗), dimension(:), intent(in) *dnames,* character(len = ∗), intent(in) *v_name,* integer(i4), intent(in), optional *dim_unlimited,* character(len = ∗), intent(in), optional *long_name,* character(len = ∗), intent(in), optional *units,* real(dp), intent(in), optional *missing_value,* character(256), dimension(:, :), intent(in), optional *attributes,* logical, intent(in), optional *create,* integer(i4), intent(inout), optional *ncid,* integer(i4), intent(in), optional *nrec*) [private]

Definition at line 3100 of file mo_ncwrite.f90.

References mo_ncread::check(), close_netcdf(), and open_netcdf().

Here is the call graph for this function:

15.112.2.21 subroutine mo_ncwrite::var2nc_1d_i4 (character(len = ∗), intent(in) *f_name,* integer(i4), dimension(:), intent(in) *arr,* character(len = ∗), dimension(:), intent(in) *dnames,* character(len = ∗), intent(in) *v_name,* integer(i4), intent(in), optional *dim_unlimited,* character(len = ∗), intent(in), optional *long_name,* character(len = ∗), intent(in), optional *units,* integer(i4), intent(in), optional *missing_value,* character(256), dimension(:, :), intent(in), optional *attributes,* logical, intent(in), optional *create,* integer(i4), intent(inout), optional *ncid,* integer(i4), intent(in), optional *nrec*) [private]

Definition at line 2759 of file mo_ncwrite.f90.

References mo_ncread::check(), close_netcdf(), and open_netcdf().

Here is the call graph for this function:

15.112.2.22 subroutine mo_ncwrite::var2nc_1d_sp (character(len = ∗), intent(in) *f_name,* real(sp), dimension(:), intent(in) *arr,* character(len = ∗), dimension(:), intent(in) *dnames,* character(len = ∗), intent(in) *v_name,* integer(i4), intent(in), optional *dim_unlimited,* character(len = ∗), intent(in), optional *long_name,* character(len = ∗), intent(in), optional *units,* real(sp), intent(in), optional *missing_value,* character(256), dimension(:, :), intent(in), optional *attributes,* logical, intent(in), optional *create,* integer(i4), intent(inout), optional *ncid,* integer(i4), intent(in), optional *nrec*) [private]

Definition at line 2930 of file mo_ncwrite.f90.

References mo_ncread::check(), close_netcdf(), and open_netcdf().

Here is the call graph for this function:

15.112.2.23 **subroutine mo_ncwrite::var2nc_2d_dp (character(len = ∗), intent(in) *f_name,* real(dp), dimension(:, :), intent(in) *arr,* character(len = ∗), dimension(:), intent(in) *dnames,* character(len = ∗), intent(in) *v_name,* integer(i4), intent(in), optional *dim_unlimited,* character(len = ∗), intent(in), optional *long_name,* character(len = ∗), intent(in), optional *units,* real(dp), intent(in), optional *missing_value,* character(256), dimension(:, :), intent(in), optional *attributes,* logical, intent(in), optional *create,* integer(i4), intent(inout), optional *ncid,* integer(i4), intent(in), optional *nrec*)** `[private]`

Definition at line 3630 of file mo_ncwrite.f90.

References mo_ncread::check(), close_netcdf(), and open_netcdf().

Here is the call graph for this function:

15.112.2.24 **subroutine mo_ncwrite::var2nc_2d_i4 (character(len = ∗), intent(in) *f_name,* integer(i4), dimension(:, :), intent(in) *arr,* character(len = ∗), dimension(:), intent(in) *dnames,* character(len = ∗), intent(in) *v_name,* integer(i4), intent(in), optional *dim_unlimited,* character(len = ∗), intent(in), optional *long_name,* character(len = ∗), intent(in), optional *units,* integer(i4), intent(in), optional *missing_value,* character(256), dimension(:, :), intent(in), optional *attributes,* logical, intent(in), optional *create,* integer(i4), intent(inout), optional *ncid,* integer(i4), intent(in), optional *nrec*)** `[private]`

Definition at line 3270 of file mo_ncwrite.f90.

References mo_ncread::check(), close_netcdf(), and open_netcdf().

Here is the call graph for this function:

15.112.2.25 **subroutine mo_ncwrite::var2nc_2d_sp (character(len = ∗), intent(in) *f_name,* real(sp), dimension(:, :), intent(in) *arr,* character(len = ∗), dimension(:), intent(in) *dnames,* character(len = ∗), intent(in) *v_name,* integer(i4), intent(in), optional *dim_unlimited,* character(len = ∗), intent(in), optional *long_name,* character(len = ∗), intent(in), optional *units,* real(sp), intent(in), optional *missing_value,* character(256), dimension(:, :), intent(in), optional *attributes,* logical, intent(in), optional *create,* integer(i4), intent(inout), optional *ncid,* integer(i4), intent(in), optional *nrec*)** `[private]`

Definition at line 3450 of file mo_ncwrite.f90.

References mo_ncread::check(), close_netcdf(), and open_netcdf().

Here is the call graph for this function:

15.112.2.26 **subroutine mo_ncwrite::var2nc_3d_dp (character(len = ∗), intent(in) *f_name,* real(dp), dimension(:, :, :), intent(in) *arr,* character(len = ∗), dimension(:), intent(in) *dnames,* character(len = ∗), intent(in) *v_name,* integer(i4), intent(in), optional *dim_unlimited,* character(len = ∗), intent(in), optional *long_name,* character(len = ∗), intent(in), optional *units,* real(dp), intent(in), optional *missing_value,* character(256), dimension(:, :), intent(in), optional *attributes,* logical, intent(in), optional *create,* integer(i4), intent(inout), optional *ncid,* integer(i4), intent(in), optional *nrec*)** `[private]`

Definition at line 4171 of file mo_ncwrite.f90.

References mo_ncread::check(), close_netcdf(), and open_netcdf().

Here is the call graph for this function:

15.112.2.27 **subroutine mo_ncwrite::var2nc_3d_i4 (character(len = ∗), intent(in) *f_name,* integer(i4), dimension(:, :, :), intent(in) *arr,* character(len = ∗), dimension(:), intent(in) *dnames,* character(len = ∗), intent(in) *v_name,* integer(i4), intent(in), optional *dim_unlimited,* character(len = ∗), intent(in), optional *long_name,* character(len = ∗), intent(in), optional *units,* integer(i4), intent(in), optional *missing_value,* character(256), dimension(:, :), intent(in), optional *attributes,* logical, intent(in), optional *create,* integer(i4), intent(inout), optional *ncid,* integer(i4), intent(in), optional *nrec*)** `[private]`

Definition at line 3810 of file mo_ncwrite.f90.

References mo_ncread::check(), close_netcdf(), and open_netcdf().

Here is the call graph for this function:

15.112.2.28 **subroutine mo_ncwrite::var2nc_3d_sp (character(len = ∗), intent(in) *f_name,* real(sp), dimension(:, :, :), intent(in) *arr,* character(len = ∗), dimension(:), intent(in) *dnames,* character(len = ∗), intent(in) *v_name,* integer(i4), intent(in), optional *dim_unlimited,* character(len = ∗), intent(in), optional *long_name,* character(len = ∗), intent(in), optional *units,* real(sp), intent(in), optional *missing_value,* character(256), dimension(:, :), intent(in), optional *attributes,* logical, intent(in), optional *create,* integer(i4), intent(inout), optional *ncid,* integer(i4), intent(in), optional *nrec*)** `[private]`

Definition at line 3990 of file mo_ncwrite.f90.

References mo_ncread::check(), close_netcdf(), and open_netcdf().

Here is the call graph for this function:

15.112.2.29 **subroutine mo_ncwrite::var2nc_4d_dp (character(len = ∗), intent(in) *f_name,* real(dp), dimension(:, :, :, :), intent(in) *arr,* character(len = ∗), dimension(:), intent(in) *dnames,* character(len = ∗), intent(in) *v_name,* integer(i4), intent(in), optional *dim_unlimited,* character(len = ∗), intent(in), optional *long_name,* character(len = ∗), intent(in), optional *units,* real(dp), intent(in), optional *missing_value,* character(256), dimension(:, :), intent(in), optional *attributes,* logical, intent(in), optional *create,* integer(i4), intent(inout), optional *ncid,* integer(i4), intent(in), optional *nrec*)** `[private]`

Definition at line 4715 of file mo_ncwrite.f90.

References mo_ncread::check(), close_netcdf(), and open_netcdf().

Here is the call graph for this function:

15.112.2.30 **subroutine mo_ncwrite::var2nc_4d_i4 (character(len = ∗), intent(in) *f_name,* integer(i4), dimension(:, :, :, :), intent(in) *arr,* character(len = ∗), dimension(:), intent(in) *dnames,* character(len = ∗), intent(in) *v_name,* integer(i4), intent(in), optional *dim_unlimited,* character(len = ∗), intent(in), optional *long_name,* character(len = ∗), intent(in), optional *units,* integer(i4), intent(in), optional *missing_value,* character(256), dimension(:, :), intent(in), optional *attributes,* logical, intent(in), optional *create,* integer(i4), intent(inout), optional *ncid,* integer(i4), intent(in), optional *nrec*)** `[private]`

Definition at line 4351 of file mo_ncwrite.f90.

References mo_ncread::check(), close_netcdf(), and open_netcdf().

Here is the call graph for this function:

15.112.2.31 **subroutine mo_ncwrite::var2nc_4d_sp (character(len = ∗), intent(in) *f_name,* real(sp), dimension(:, :, :, :), intent(in) *arr,* character(len = ∗), dimension(:), intent(in) *dnames,* character(len = ∗), intent(in) *v_name,* integer(i4), intent(in), optional *dim_unlimited,* character(len = ∗), intent(in), optional *long_name,* character(len = ∗), intent(in), optional *units,* real(sp), intent(in), optional *missing_value,* character(256), dimension(:, :), intent(in), optional *attributes,* logical, intent(in), optional *create,* integer(i4), intent(inout), optional *ncid,* integer(i4), intent(in), optional *nrec*)** `[private]`

Definition at line 4533 of file mo_ncwrite.f90.

References mo_ncread::check(), close_netcdf(), and open_netcdf().

Here is the call graph for this function:

15.112.2.32 **subroutine mo_ncwrite::var2nc_5d_dp (character(len = ∗), intent(in) *f_name,* real(dp), dimension(:, :, :, :, :), intent(in) *arr,* character(len = ∗), dimension(:), intent(in) *dnames,* character(len = ∗), intent(in) *v_name,* integer(i4), intent(in), optional *dim_unlimited,* character(len = ∗), intent(in), optional *long_name,* character(len = ∗), intent(in), optional *units,* real(dp), intent(in), optional *missing_value,* character(256), dimension(:, :), intent(in), optional *attributes,* logical, intent(in), optional *create,* integer(i4), intent(inout), optional *ncid,* integer(i4), intent(in), optional *nrec*)** `[private]`

Definition at line 5241 of file mo_ncwrite.f90.

References mo_ncread::check(), close_netcdf(), and open_netcdf().

Here is the call graph for this function:

15.112.2.33 **subroutine mo_ncwrite::var2nc_5d_i4 (character(len = ∗), intent(in) *f_name,* integer(i4), dimension(:, :, :, :, :), intent(in) *arr,* character(len = ∗), dimension(:), intent(in) *dnames,* character(len = ∗), intent(in) *v_name,* integer(i4), intent(in), optional *dim_unlimited,* character(len = ∗), intent(in), optional *long_name,* character(len = ∗), intent(in), optional *units,* integer(i4), intent(in), optional *missing_value,* character(256), dimension(:, :), intent(in), optional *attributes,* logical, intent(in), optional *create,* integer(i4), intent(inout), optional *ncid,* integer(i4), intent(in), optional *nrec*)** `[private]`

Definition at line 4897 of file mo_ncwrite.f90.

References mo_ncread::check(), close_netcdf(), and open_netcdf().

Here is the call graph for this function:

15.112.2.34 **subroutine mo_ncwrite::var2nc_5d_sp (character(len = ∗), intent(in) *f_name,* real(sp), dimension(:, :, :, :, :), intent(in) *arr,* character(len = ∗), dimension(:), intent(in) *dnames,* character(len = ∗), intent(in) *v_name,* integer(i4), intent(in), optional *dim_unlimited,* character(len = ∗), intent(in), optional *long_name,* character(len = ∗), intent(in), optional *units,* real(sp), intent(in), optional *missing_value,* character(256), dimension(:, :), intent(in), optional *attributes,* logical, intent(in), optional *create,* integer(i4), intent(inout), optional *ncid,* integer(i4), intent(in), optional *nrec*)** `[private]`

Definition at line 5069 of file mo_ncwrite.f90.

References mo_ncread::check(), close_netcdf(), and open_netcdf().

Here is the call graph for this function:

15.112.2.35 subroutine, public mo_ncwrite::write_dynamic_netcdf (integer(i4), intent(in) *ncId,* integer(i4), intent(in) *irec*)

Definition at line 5452 of file mo_ncwrite.f90.

References mo_ncread::check().

Here is the call graph for this function:

15.112.2.36 subroutine, public mo_ncwrite::write_static_netcdf (integer(i4), intent(in) *ncId*)

Definition at line 5561 of file mo_ncwrite.f90.

References mo_ncread::check().

Here is the call graph for this function:

15.112.3 Member Data Documentation

15.112.3.1 type (dims), dimension(:), allocatable, public mo_ncwrite::dnc

Definition at line 112 of file mo_ncwrite.f90.

15.112.3.2 type(attribute), dimension(ngatt), public mo_ncwrite::gatt

Definition at line 114 of file mo_ncwrite.f90.

15.112.3.3 integer(i4), parameter, public mo_ncwrite::maxlen = 256

Definition at line 54 of file mo_ncwrite.f90.

15.112.3.4 integer(i4), parameter, public mo_ncwrite::nattdim = 2

Definition at line 56 of file mo_ncwrite.f90.

15.112.3.5 integer(i4), public mo_ncwrite::ndims

Definition at line 111 of file mo_ncwrite.f90.

15.112.3.6 integer(i4), parameter, public mo_ncwrite::ngatt = 20

Definition at line 55 of file mo_ncwrite.f90.

15.112.3.7 integer(i4), parameter, public mo_ncwrite::nmaxatt = 20

Definition at line 53 of file mo_ncwrite.f90.

15.112.3.8 integer(i4), parameter, public mo_ncwrite::nmaxdim = 5

Definition at line 52 of file mo_ncwrite.f90.

15.112.3.9 integer(i4), public mo_ncwrite::nvars

Definition at line 110 of file mo_ncwrite.f90.

15.112.3.10 type(variable), dimension(:), allocatable, public mo_ncwrite::v

Definition at line 113 of file mo_ncwrite.f90.

The documentation for this module was generated from the following file:

- mo_ncwrite.f90

15.113 mo_netcdf Module Reference

NetCDF Fortran 90 interface wrapper.

Data Types

- interface ncdataset

  Provides basic file modification functionality.
- type ncdimension

  Provides the dimension access functionality.
- interface ncvariable

Public Member Functions

- subroutine initncvariable (self, id, parent)
- subroutine initncdimension (self, id, parent)
- subroutine initncdataset (self, fname, mode)
- type(ncvariable) function newncvariable (id, parent)
- type(ncdimension) function newncdimension (id, parent)
- type(ncdataset) function newncdataset (fname, mode)
- subroutine close (self)
- integer(i4) function getnovariables (self)
- integer(i4) function,
  dimension(:), allocatable getvariableids (self)
- type(ncvariable) function,
  dimension(:), allocatable getvariables (self)
- character(len=256) function getdimensionname (self)
- integer(i4) function getdimensionlength (self)
- logical function isdatasetunlimited (self)
- type(ncdimension) function getunlimiteddimension (self)
- logical function equalncdimensions (dim1, dim2)
- logical function isunlimiteddimension (self)
- type(ncdimension) function setdimension (self, name, length)
- logical function hasvariable (self, name)
- logical function hasdimension (self, name)
- type(ncvariable) function setvariablewithids (self, name, dtype, dimensions, contiguous, chunksizes, deflate↩_level, shuffle, fletcher32, endianness, cache_size, cache_nelems, cache_preemption)
- type(ncvariable) function setvariablewithnames (self, name, dtype, dimensions, contiguous, chunksizes, deflate_level, shuffle, fletcher32, endianness, cache_size, cache_nelems, cache_preemption)
- type(ncvariable) function setvariablewithtypes (self, name, dtype, dimensions, contiguous, chunksizes, deflate_level, shuffle, fletcher32, endianness, cache_size, cache_nelems, cache_preemption)
- type(ncdimension) function getdimensionbyid (self, id)
- type(ncdimension) function getdimensionbyname (self, name)
- type(ncvariable) function getvariablebyname (self, name)
- character(len=256) function getvariablename (self)
- integer(i4) function getnodimensions (self)
- type(ncdimension) function,
  dimension(:), allocatable getvariabledimensions (self)
- integer(i4) function,
  dimension(:), allocatable getvariableshape (self)
- character(3) function getvariabledtype (self)
- logical function isunlimitedvariable (self)
- logical function hasattribute (self, name)
- subroutine setglobalattributechar (self, name, data)
- subroutine setglobalattributei8 (self, name, data)
- subroutine setglobalattributei16 (self, name, data)
- subroutine setglobalattributei32 (self, name, data)
- subroutine setglobalattributei64 (self, name, data)

- subroutine setglobalattributef32 (self, name, data)
- subroutine setglobalattributef64 (self, name, data)
- subroutine getglobalattributechar (self, name, avalue)
- subroutine getglobalattributei8 (self, name, avalue)
- subroutine getglobalattributei16 (self, name, avalue)
- subroutine getglobalattributei32 (self, name, avalue)
- subroutine getglobalattributei64 (self, name, avalue)
- subroutine getglobalattributef32 (self, name, avalue)
- subroutine getglobalattributef64 (self, name, avalue)
- subroutine setvariableattributechar (self, name, data)
- subroutine setvariableattributei8 (self, name, data)
- subroutine setvariableattributei16 (self, name, data)
- subroutine setvariableattributei32 (self, name, data)
- subroutine setvariableattributei64 (self, name, data)
- subroutine setvariableattributef32 (self, name, data)
- subroutine setvariableattributef64 (self, name, data)
- subroutine getvariableattributechar (self, name, avalue)
- subroutine getvariableattributei8 (self, name, avalue)
- subroutine getvariableattributei16 (self, name, avalue)
- subroutine getvariableattributei32 (self, name, avalue)
- subroutine getvariableattributei64 (self, name, avalue)
- subroutine getvariableattributef32 (self, name, avalue)
- subroutine getvariableattributef64 (self, name, avalue)
- subroutine setvariablefillvaluei8 (self, fvalue)
- subroutine setvariablefillvaluei16 (self, fvalue)
- subroutine setvariablefillvaluei32 (self, fvalue)
- subroutine setvariablefillvaluei64 (self, fvalue)
- subroutine setvariablefillvaluef32 (self, fvalue)
- subroutine setvariablefillvaluef64 (self, fvalue)
- subroutine getvariablefillvaluei8 (self, fvalue)
- subroutine getvariablefillvaluei16 (self, fvalue)
- subroutine getvariablefillvaluei32 (self, fvalue)
- subroutine getvariablefillvaluei64 (self, fvalue)
- subroutine getvariablefillvaluef32 (self, fvalue)
- subroutine getvariablefillvaluef64 (self, fvalue)
- subroutine setdatascalari8 (self, values, start)
- subroutine setdata1di8 (self, values, start, cnt, stride, map)
- subroutine setdata2di8 (self, values, start, cnt, stride, map)
- subroutine setdata3di8 (self, values, start, cnt, stride, map)
- subroutine setdata4di8 (self, values, start, cnt, stride, map)
- subroutine setdata5di8 (self, values, start, cnt, stride, map)
- subroutine setdatascalari16 (self, values, start)
- subroutine setdata1di16 (self, values, start, cnt, stride, map)
- subroutine setdata2di16 (self, values, start, cnt, stride, map)
- subroutine setdata3di16 (self, values, start, cnt, stride, map)
- subroutine setdata4di16 (self, values, start, cnt, stride, map)
- subroutine setdata5di16 (self, values, start, cnt, stride, map)
- subroutine setdatascalari32 (self, values, start)
- subroutine setdata1di32 (self, values, start, cnt, stride, map)
- subroutine setdata2di32 (self, values, start, cnt, stride, map)
- subroutine setdata3di32 (self, values, start, cnt, stride, map)
- subroutine setdata4di32 (self, values, start, cnt, stride, map)
- subroutine setdata5di32 (self, values, start, cnt, stride, map)
- subroutine setdatascalari64 (self, values, start)
- subroutine setdata1di64 (self, values, start, cnt, stride, map)

- subroutine setdata2di64 (self, values, start, cnt, stride, map)
- subroutine setdata3di64 (self, values, start, cnt, stride, map)
- subroutine setdata4di64 (self, values, start, cnt, stride, map)
- subroutine setdata5di64 (self, values, start, cnt, stride, map)
- subroutine setdatascalarf32 (self, values, start)
- subroutine setdata1df32 (self, values, start, cnt, stride, map)
- subroutine setdata2df32 (self, values, start, cnt, stride, map)
- subroutine setdata3df32 (self, values, start, cnt, stride, map)
- subroutine setdata4df32 (self, values, start, cnt, stride, map)
- subroutine setdata5df32 (self, values, start, cnt, stride, map)
- subroutine setdatascalarf64 (self, values, start)
- subroutine setdata1df64 (self, values, start, cnt, stride, map)
- subroutine setdata2df64 (self, values, start, cnt, stride, map)
- subroutine setdata3df64 (self, values, start, cnt, stride, map)
- subroutine setdata4df64 (self, values, start, cnt, stride, map)
- subroutine setdata5df64 (self, values, start, cnt, stride, map)
- subroutine getdatascalari8 (self, data, start, cnt, stride, map)
- subroutine getdata1di8 (self, data, start, cnt, stride, map)
- subroutine getdata2di8 (self, data, start, cnt, stride, map)
- subroutine getdata3di8 (self, data, start, cnt, stride, map)
- subroutine getdata4di8 (self, data, start, cnt, stride, map)
- subroutine getdata5di8 (self, data, start, cnt, stride, map)
- subroutine getdatascalari16 (self, data, start, cnt, stride, map)
- subroutine getdata1di16 (self, data, start, cnt, stride, map)
- subroutine getdata2di16 (self, data, start, cnt, stride, map)
- subroutine getdata3di16 (self, data, start, cnt, stride, map)
- subroutine getdata4di16 (self, data, start, cnt, stride, map)
- subroutine getdata5di16 (self, data, start, cnt, stride, map)
- subroutine getdatascalari32 (self, data, start, cnt, stride, map)
- subroutine getdata1di32 (self, data, start, cnt, stride, map)
- subroutine getdata2di32 (self, data, start, cnt, stride, map)
- subroutine getdata3di32 (self, data, start, cnt, stride, map)
- subroutine getdata4di32 (self, data, start, cnt, stride, map)
- subroutine getdata5di32 (self, data, start, cnt, stride, map)
- subroutine getdatascalari64 (self, data, start, cnt, stride, map)
- subroutine getdata1di64 (self, data, start, cnt, stride, map)
- subroutine getdata2di64 (self, data, start, cnt, stride, map)
- subroutine getdata3di64 (self, data, start, cnt, stride, map)
- subroutine getdata4di64 (self, data, start, cnt, stride, map)
- subroutine getdata5di64 (self, data, start, cnt, stride, map)
- subroutine getdatascalarf32 (self, data, start, cnt, stride, map)
- subroutine getdata1df32 (self, data, start, cnt, stride, map)
- subroutine getdata2df32 (self, data, start, cnt, stride, map)
- subroutine getdata3df32 (self, data, start, cnt, stride, map)
- subroutine getdata4df32 (self, data, start, cnt, stride, map)
- subroutine getdata5df32 (self, data, start, cnt, stride, map)
- subroutine getdatascalarf64 (self, data, start, cnt, stride, map)
- subroutine getdata1df64 (self, data, start, cnt, stride, map)
- subroutine getdata2df64 (self, data, start, cnt, stride, map)
- subroutine getdata3df64 (self, data, start, cnt, stride, map)
- subroutine getdata4df64 (self, data, start, cnt, stride, map)
- subroutine getdata5df64 (self, data, start, cnt, stride, map)
- integer(i4) function,
  dimension(datarank) getreaddatashape (var, datarank, instart, incnt, instride)
- integer(i4) function getdtypefromstring (dtype)
- character(3) function getdtypefrominteger (dtype)
- subroutine check (status, msg)

15.113.1 Detailed Description

NetCDF Fortran 90 interface wrapper.

A thin wrapper around the NetCDF Fortran 90 interface. Provided are currently 3 user facing derived Types:

1. NcDataset
2. NcDimension
3. NcVariable

Authors

David Schaefer

Date

Jun 2015

Definition at line 15 of file mo_netcdf.f90.

15.113.2 Member Function/Subroutine Documentation

15.113.2.1 subroutine mo_netcdf::check (integer(i4), intent(in) *status,* character(∗), intent(in) *msg*)

Definition at line 3041 of file mo_netcdf.f90.

15.113.2.2 subroutine mo_netcdf::close (class(ncdataset) *self*)

Definition at line 1404 of file mo_netcdf.f90.

References mo_ncread::check().

Here is the call graph for this function:

15.113.2.3 logical function mo_netcdf::equalncdimensions (type(ncdimension), intent(in) *dim1,* type(ncdimension), intent(in) *dim2*)

Definition at line 1484 of file mo_netcdf.f90.

15.113.2.4 subroutine mo_netcdf::getdata1df32 (class(ncvariable), intent(in) *self,* real(sp), dimension(:), intent(out), allocatable *data,* integer(i4), dimension(:), intent(in), optional *start,* integer(i4), dimension(:), intent(in), optional *cnt,* integer(i4), dimension(:), intent(in), optional *stride,* integer(i4), dimension(:), intent(in), optional *map*)

Definition at line 2802 of file mo_netcdf.f90.

References mo_ncread::check(), and getreaddatashape().

Here is the call graph for this function:

15.113.2.5 subroutine mo_netcdf::getdata1df64 (class(ncvariable), intent(in) *self,* real(dp), dimension(:), intent(out), allocatable *data,* integer(i4), dimension(:), intent(in), optional *start,* integer(i4), dimension(:), intent(in), optional *cnt,* integer(i4), dimension(:), intent(in), optional *stride,* integer(i4), dimension(:), intent(in), optional *map*)

Definition at line 2883 of file mo_netcdf.f90.

References mo_ncread::check(), and getreaddatashape().

Here is the call graph for this function:

15.113.2.6 subroutine mo_netcdf::getdata1di16 (class(ncvariable), intent(in) *self,* integer(i2), dimension(:), intent(out), allocatable *data,* integer(i4), dimension(:), intent(in), optional *start,* integer(i4), dimension(:), intent(in), optional *cnt,* integer(i4), dimension(:), intent(in), optional *stride,* integer(i4), dimension(:), intent(in), optional *map*)

Definition at line 2559 of file mo_netcdf.f90.

References mo_ncread::check(), and getreaddatashape().

Here is the call graph for this function:

15.113.2.7 subroutine mo_netcdf::getdata1di32 (class(ncvariable), intent(in) *self,* integer(i4), dimension(:), intent(out), allocatable *data,* integer(i4), dimension(:), intent(in), optional *start,* integer(i4), dimension(:), intent(in), optional *cnt,* integer(i4), dimension(:), intent(in), optional *stride,* integer(i4), dimension(:), intent(in), optional *map*)

Definition at line 2640 of file mo_netcdf.f90.

References mo_ncread::check(), and getreaddatashape().

Here is the call graph for this function:

15.113.2.8 subroutine mo_netcdf::getdata1di64 (class(ncvariable), intent(in) *self,* integer(i8), dimension(:), intent(out), allocatable *data,* integer(i4), dimension(:), intent(in), optional *start,* integer(i4), dimension(:), intent(in), optional *cnt,* integer(i4), dimension(:), intent(in), optional *stride,* integer(i4), dimension(:), intent(in), optional *map*)

Definition at line 2721 of file mo_netcdf.f90.

References mo_ncread::check(), and getreaddatashape().

Here is the call graph for this function:

15.113.2.9 subroutine mo_netcdf::getdata1di8 (class(ncvariable), intent(in) *self,* integer(i1), dimension(:), intent(out), allocatable *data,* integer(i4), dimension(:), intent(in), optional *start,* integer(i4), dimension(:), intent(in), optional *cnt,* integer(i4), dimension(:), intent(in), optional *stride,* integer(i4), dimension(:), intent(in), optional *map*)

Definition at line 2478 of file mo_netcdf.f90.

References mo_ncread::check(), and getreaddatashape().

Here is the call graph for this function:

15.113.2.10 subroutine mo_netcdf::getdata2df32 (class(ncvariable), intent(in) *self,* real(sp), dimension(:, :), intent(out), allocatable *data,* integer(i4), dimension(:), intent(in), optional *start,* integer(i4), dimension(:), intent(in), optional *cnt,* integer(i4), dimension(:), intent(in), optional *stride,* integer(i4), dimension(:), intent(in), optional *map*)

Definition at line 2816 of file mo_netcdf.f90.

References mo_ncread::check(), and getreaddatashape().

Here is the call graph for this function:

15.113.2.11 **subroutine mo_netcdf::getdata2df64 (class(ncvariable), intent(in) *self,* real(dp), dimension(:, :), intent(out), allocatable *data,* integer(i4), dimension(:), intent(in), optional *start,* integer(i4), dimension(:), intent(in), optional *cnt,* integer(i4), dimension(:), intent(in), optional *stride,* integer(i4), dimension(:), intent(in), optional *map*)**

Definition at line 2897 of file mo_netcdf.f90.

References mo_ncread::check(), and getreaddatashape().

Here is the call graph for this function:

15.113.2.12 **subroutine mo_netcdf::getdata2di16 (class(ncvariable), intent(in) *self,* integer(i2), dimension(:, :), intent(out), allocatable *data,* integer(i4), dimension(:), intent(in), optional *start,* integer(i4), dimension(:), intent(in), optional *cnt,* integer(i4), dimension(:), intent(in), optional *stride,* integer(i4), dimension(:), intent(in), optional *map*)**

Definition at line 2573 of file mo_netcdf.f90.

References mo_ncread::check(), and getreaddatashape().

Here is the call graph for this function:

15.113.2.13 subroutine mo_netcdf::getdata2di32 (class(ncvariable), intent(in) *self,* integer(i4), dimension(:, :), intent(out), allocatable *data,* integer(i4), dimension(:), intent(in), optional *start,* integer(i4), dimension(:), intent(in), optional *cnt,* integer(i4), dimension(:), intent(in), optional *stride,* integer(i4), dimension(:), intent(in), optional *map*)

Definition at line 2654 of file mo_netcdf.f90.

References mo_ncread::check(), and getreaddatashape().

Here is the call graph for this function:

15.113.2.14 subroutine mo_netcdf::getdata2di64 (class(ncvariable), intent(in) *self,* integer(i8), dimension(:, :), intent(out), allocatable *data,* integer(i4), dimension(:), intent(in), optional *start,* integer(i4), dimension(:), intent(in), optional *cnt,* integer(i4), dimension(:), intent(in), optional *stride,* integer(i4), dimension(:), intent(in), optional *map*)

Definition at line 2735 of file mo_netcdf.f90.

References mo_ncread::check(), and getreaddatashape().

Here is the call graph for this function:

15.113.2.15 subroutine mo_netcdf::getdata2di8 (class(ncvariable), intent(in) *self,* integer(i1), dimension(:, :), intent(out), allocatable *data,* integer(i4), dimension(:), intent(in), optional *start,* integer(i4), dimension(:), intent(in), optional *cnt,* integer(i4), dimension(:), intent(in), optional *stride,* integer(i4), dimension(:), intent(in), optional *map*)

Definition at line 2492 of file mo_netcdf.f90.

References mo_ncread::check(), and getreaddatashape().

Here is the call graph for this function:

15.113.2.16 subroutine mo_netcdf::getdata3df32 (class(ncvariable), intent(in) *self,* real(sp), dimension(:, :, :), intent(out), allocatable *data,* integer(i4), dimension(:), intent(in), optional *start,* integer(i4), dimension(:), intent(in), optional *cnt,* integer(i4), dimension(:), intent(in), optional *stride,* integer(i4), dimension(:), intent(in), optional *map*)

Definition at line 2830 of file mo_netcdf.f90.

References mo_ncread::check(), and getreaddatashape().

Here is the call graph for this function:

15.113.2.17 **subroutine mo_netcdf::getdata3df64 (class(ncvariable), intent(in) *self,* real(dp), dimension(:, :, :), intent(out), allocatable *data,* integer(i4), dimension(:), intent(in), optional *start,* integer(i4), dimension(:), intent(in), optional *cnt,* integer(i4), dimension(:), intent(in), optional *stride,* integer(i4), dimension(:), intent(in), optional *map*)**

Definition at line 2911 of file mo_netcdf.f90.

References mo_ncread::check(), and getreaddatashape().

Here is the call graph for this function:

15.113.2.18 **subroutine mo_netcdf::getdata3di16 (class(ncvariable), intent(in) *self,* integer(i2), dimension(:, :, :), intent(out), allocatable *data,* integer(i4), dimension(:), intent(in), optional *start,* integer(i4), dimension(:), intent(in), optional *cnt,* integer(i4), dimension(:), intent(in), optional *stride,* integer(i4), dimension(:), intent(in), optional *map*)**

Definition at line 2587 of file mo_netcdf.f90.

References mo_ncread::check(), and getreaddatashape().

Here is the call graph for this function:

15.113.2.19 **subroutine mo_netcdf::getdata3di32 (class(ncvariable), intent(in) *self,* integer(i4), dimension(:, :, :), intent(out), allocatable *data,* integer(i4), dimension(:), intent(in), optional *start,* integer(i4), dimension(:), intent(in), optional *cnt,* integer(i4), dimension(:), intent(in), optional *stride,* integer(i4), dimension(:), intent(in), optional *map*)**

Definition at line 2668 of file mo_netcdf.f90.

References mo_ncread::check(), and getreaddatashape().

Here is the call graph for this function:

15.113.2.20 **subroutine mo_netcdf::getdata3di64 (class(ncvariable), intent(in) *self,* integer(i8), dimension(:, :, :), intent(out), allocatable *data,* integer(i4), dimension(:), intent(in), optional *start,* integer(i4), dimension(:), intent(in), optional *cnt,* integer(i4), dimension(:), intent(in), optional *stride,* integer(i4), dimension(:), intent(in), optional *map*)**

Definition at line 2749 of file mo_netcdf.f90.

References mo_ncread::check(), and getreaddatashape().

Here is the call graph for this function:

15.113.2.21 subroutine mo_netcdf::getdata3di8 (class(**ncvariable**), intent(in) *self,* integer(i1), dimension(:, :, :), intent(out), allocatable *data,* integer(i4), dimension(:), intent(in), optional *start,* integer(i4), dimension(:), intent(in), optional *cnt,* integer(i4), dimension(:), intent(in), optional *stride,* integer(i4), dimension(:), intent(in), optional *map*)

Definition at line 2506 of file mo_netcdf.f90.

References mo_ncread::check(), and getreaddatashape().

Here is the call graph for this function:

15.113.2.22 subroutine mo_netcdf::getdata4df32 (class(**ncvariable**), intent(in) *self,* real(sp), dimension(:, :, :, :), intent(out), allocatable *data,* integer(i4), dimension(:), intent(in), optional *start,* integer(i4), dimension(:), intent(in), optional *cnt,* integer(i4), dimension(:), intent(in), optional *stride,* integer(i4), dimension(:), intent(in), optional *map*)

Definition at line 2844 of file mo_netcdf.f90.

References mo_ncread::check(), and getreaddatashape().

Here is the call graph for this function:

15.113.2.23 subroutine mo_netcdf::getdata4df64 (class(**ncvariable**), intent(in) *self,* real(dp), dimension(:, :, :, :), intent(out), allocatable *data,* integer(i4), dimension(:), intent(in), optional *start,* integer(i4), dimension(:), intent(in), optional *cnt,* integer(i4), dimension(:), intent(in), optional *stride,* integer(i4), dimension(:), intent(in), optional *map*)

Definition at line 2925 of file mo_netcdf.f90.

References mo_ncread::check(), and getreaddatashape().

Here is the call graph for this function:

15.113.2.24 subroutine mo_netcdf::getdata4di16 (class(**ncvariable**), intent(in) *self,* integer(i2), dimension(:, :, :, :), intent(out), allocatable *data,* integer(i4), dimension(:), intent(in), optional *start,* integer(i4), dimension(:), intent(in), optional *cnt,* integer(i4), dimension(:), intent(in), optional *stride,* integer(i4), dimension(:), intent(in), optional *map*)

Definition at line 2601 of file mo_netcdf.f90.

References mo_ncread::check(), and getreaddatashape().

Here is the call graph for this function:

15.113.2.25 subroutine mo_netcdf::getdata4di32 (class(**ncvariable**), intent(in) *self,* integer(i4), dimension(:, :, :, :), intent(out), allocatable *data,* integer(i4), dimension(:), intent(in), optional *start,* integer(i4), dimension(:), intent(in), optional *cnt,* integer(i4), dimension(:), intent(in), optional *stride,* integer(i4), dimension(:), intent(in), optional *map*)

Definition at line 2682 of file mo_netcdf.f90.

References mo_ncread::check(), and getreaddatashape().

Here is the call graph for this function:

15.113.2.26 subroutine mo_netcdf::getdata4di64 (class(**ncvariable**), intent(in) *self,* integer(i8), dimension(:, :, :, :), intent(out), allocatable *data,* integer(i4), dimension(:), intent(in), optional *start,* integer(i4), dimension(:), intent(in), optional *cnt,* integer(i4), dimension(:), intent(in), optional *stride,* integer(i4), dimension(:), intent(in), optional *map*)

Definition at line 2763 of file mo_netcdf.f90.

References mo_ncread::check(), and getreaddatashape().

Here is the call graph for this function:

15.113.2.27 subroutine mo_netcdf::getdata4di8 (class(**ncvariable**), intent(in) *self,* integer(i1), dimension(:, :, :, :), intent(out), allocatable *data,* integer(i4), dimension(:), intent(in), optional *start,* integer(i4), dimension(:), intent(in), optional *cnt,* integer(i4), dimension(:), intent(in), optional *stride,* integer(i4), dimension(:), intent(in), optional *map*)

Definition at line 2520 of file mo_netcdf.f90.

References mo_ncread::check(), and getreaddatashape().

Here is the call graph for this function:

15.113.2.28 subroutine mo_netcdf::getdata5df32 (class(**ncvariable**), intent(in) *self,* real(sp), dimension(:, :, :, :, :), intent(out), allocatable *data,* integer(i4), dimension(:), intent(in), optional *start,* integer(i4), dimension(:), intent(in), optional *cnt,* integer(i4), dimension(:), intent(in), optional *stride,* integer(i4), dimension(:), intent(in), optional *map*)

Definition at line 2858 of file mo_netcdf.f90.

References mo_ncread::check(), and getreaddatashape().

Here is the call graph for this function:

15.113.2.29 **subroutine mo_netcdf::getdata5df64 (class(ncvariable), intent(in) *self,* real(dp), dimension(:, :, :, :), intent(out), allocatable *data,* integer(i4), dimension(:), intent(in), optional *start,* integer(i4), dimension(:), intent(in), optional *cnt,* integer(i4), dimension(:), intent(in), optional *stride,* integer(i4), dimension(:), intent(in), optional *map*)**

Definition at line 2939 of file mo_netcdf.f90.

References mo_ncread::check(), and getreaddatashape().

Here is the call graph for this function:

15.113.2.30 **subroutine mo_netcdf::getdata5di16 (class(ncvariable), intent(in) *self,* integer(i2), dimension(:, :, :, :), intent(out), allocatable *data,* integer(i4), dimension(:), intent(in), optional *start,* integer(i4), dimension(:), intent(in), optional *cnt,* integer(i4), dimension(:), intent(in), optional *stride,* integer(i4), dimension(:), intent(in), optional *map*)**

Definition at line 2615 of file mo_netcdf.f90.

References mo_ncread::check(), and getreaddatashape().

Here is the call graph for this function:

15.113.2.31 subroutine mo_netcdf::getdata5di32 (class(**ncvariable**), intent(in) *self,* integer(i4), dimension(:, :, :, :, :), intent(out), allocatable *data,* integer(i4), dimension(:), intent(in), optional *start,* integer(i4), dimension(:), intent(in), optional *cnt,* integer(i4), dimension(:), intent(in), optional *stride,* integer(i4), dimension(:), intent(in), optional *map*)

Definition at line 2696 of file mo_netcdf.f90.

References mo_ncread::check(), and getreaddatashape().

Here is the call graph for this function:

15.113.2.32 subroutine mo_netcdf::getdata5di64 (class(**ncvariable**), intent(in) *self,* integer(i8), dimension(:, :, :, :, :), intent(out), allocatable *data,* integer(i4), dimension(:), intent(in), optional *start,* integer(i4), dimension(:), intent(in), optional *cnt,* integer(i4), dimension(:), intent(in), optional *stride,* integer(i4), dimension(:), intent(in), optional *map*)

Definition at line 2777 of file mo_netcdf.f90.

References mo_ncread::check(), and getreaddatashape().

Here is the call graph for this function:

15.113.2.33 subroutine mo_netcdf::getdata5di8 (class(ncvariable), intent(in) *self,* integer(i1), dimension(:, :, :, :, :), intent(out), allocatable *data,* integer(i4), dimension(:), intent(in), optional *start,* integer(i4), dimension(:), intent(in), optional *cnt,* integer(i4), dimension(:), intent(in), optional *stride,* integer(i4), dimension(:), intent(in), optional *map*)

Definition at line 2534 of file mo_netcdf.f90.

References mo_ncread::check(), and getreaddatashape().

Here is the call graph for this function:

15.113.2.34 subroutine mo_netcdf::getdatascalarf32 (class(ncvariable), intent(in) *self,* real(sp), intent(out) *data,* integer(i4), dimension(:), intent(in), optional *start,* integer(i4), dimension(:), intent(in), optional *cnt,* integer(i4), dimension(:), intent(in), optional *stride,* integer(i4), dimension(:), intent(in), optional *map*)

Definition at line 2791 of file mo_netcdf.f90.

References mo_ncread::check().

Here is the call graph for this function:

15.113.2.35 subroutine mo_netcdf::getdatascalarf64 (class(ncvariable), intent(in) *self,* real(dp), intent(out) *data,* integer(i4), dimension(:), intent(in), optional *start,* integer(i4), dimension(:), intent(in), optional *cnt,* integer(i4), dimension(:), intent(in), optional *stride,* integer(i4), dimension(:), intent(in), optional *map*)

Definition at line 2872 of file mo_netcdf.f90.

References mo_ncread::check().

Here is the call graph for this function:

15.113.2.36 subroutine mo_netcdf::getdatascalari16 (class(ncvariable), intent(in) *self,* integer(i2), intent(out) *data,* integer(i4), dimension(:), intent(in), optional *start,* integer(i4), dimension(:), intent(in), optional *cnt,* integer(i4), dimension(:), intent(in), optional *stride,* integer(i4), dimension(:), intent(in), optional *map*)

Definition at line 2548 of file mo_netcdf.f90.

References mo_ncread::check().

Here is the call graph for this function:

15.113.2.37 subroutine mo_netcdf::getdatascalari32 (class(ncvariable), intent(in) *self,* integer(i4), intent(out) *data,* integer(i4), dimension(:), intent(in), optional *start,* integer(i4), dimension(:), intent(in), optional *cnt,* integer(i4), dimension(:), intent(in), optional *stride,* integer(i4), dimension(:), intent(in), optional *map*)

Definition at line 2629 of file mo_netcdf.f90.

References mo_ncread::check().

Here is the call graph for this function:

15.113.2.38 subroutine mo_netcdf::getdatascalari64 (class(**ncvariable**), intent(in) *self,* integer(i8), intent(out) *data,* integer(i4), dimension(:), intent(in), optional *start,* integer(i4), dimension(:), intent(in), optional *cnt,* integer(i4), dimension(:), intent(in), optional *stride,* integer(i4), dimension(:), intent(in), optional *map*)

Definition at line 2710 of file mo_netcdf.f90.

References mo_ncread::check().

Here is the call graph for this function:

15.113.2.39 subroutine mo_netcdf::getdatascalari8 (class(**ncvariable**), intent(in) *self,* integer(i1), intent(out) *data,* integer(i4), dimension(:), intent(in), optional *start,* integer(i4), dimension(:), intent(in), optional *cnt,* integer(i4), dimension(:), intent(in), optional *stride,* integer(i4), dimension(:), intent(in), optional *map*)

Definition at line 2467 of file mo_netcdf.f90.

References mo_ncread::check().

Here is the call graph for this function:

15.113.2.40 type(**ncdimension**) function mo_netcdf::getdimensionbyid (class(**ncdataset**), intent(in) *self,* integer(i4) *id*)

Definition at line 1607 of file mo_netcdf.f90.

References mo_ncread::check(), and mo_netcdf::ncdataset::getdimensionbyid().

Here is the call graph for this function:

15.113.2.41 type(ncdimension) function mo_netcdf::getdimensionbyname (class(ncdataset), intent(in) *self*, character(∗) *name*)

Definition at line 1619 of file mo_netcdf.f90.

References mo_ncread::check(), and mo_netcdf::ncdataset::getdimensionbyname().

Here is the call graph for this function:

15.113.2.42 integer(i4) function mo_netcdf::getdimensionlength (class(ncdimension), intent(in) *self*)

Definition at line 1450 of file mo_netcdf.f90.

References mo_ncread::check().

Here is the call graph for this function:

15.113.2.43 character(len = 256) function mo_netcdf::getdimensionname (class(ncdimension), intent(in) *self*)

Definition at line 1442 of file mo_netcdf.f90.

References mo_ncread::check().

Here is the call graph for this function:

15.113.2.44 character(3) function mo_netcdf::getdtypefrominteger (integer(i4) *dtype*)

Definition at line 3018 of file mo_netcdf.f90.

Referenced by getvariabledtype().

Here is the caller graph for this function:

15.113.2.45 integer(i4) function mo_netcdf::getdtypefromstring (character(∗) *dtype*)

Definition at line 2995 of file mo_netcdf.f90.

Referenced by setvariablewithids().

Here is the caller graph for this function:

15.113.2.46 subroutine mo_netcdf::getglobalattributechar (class(ncdataset), intent(in) *self,* character(∗), intent(in) *name,* character(∗), intent(out) *avalue*)

Definition at line 1792 of file mo_netcdf.f90.

References mo_ncread::check().

Here is the call graph for this function:

15.113.2.47 subroutine mo_netcdf::getglobalattributef32 (class(ncdataset), intent(in) *self*, character(∗), intent(in) *name*, real(sp), intent(out) *avalue*)

Definition at line 1852 of file mo_netcdf.f90.

References mo_ncread::check().

Here is the call graph for this function:

15.113.2.48 subroutine mo_netcdf::getglobalattributef64 (class(ncdataset), intent(in) *self*, character(∗), intent(in) *name*, real(dp), intent(out) *avalue*)

Definition at line 1864 of file mo_netcdf.f90.

References mo_ncread::check().

Here is the call graph for this function:

15.113.2.49 subroutine mo_netcdf::getglobalattributei16 (class(ncdataset), intent(in) *self*, character(∗), intent(in) *name*, integer(i2), intent(out) *avalue*)

Definition at line 1816 of file mo_netcdf.f90.

References mo_ncread::check().

Here is the call graph for this function:

15.113.2.50 subroutine mo_netcdf::getglobalattributei32 (class(ncdataset), intent(in) *self*, character(∗), intent(in) *name*, integer(i4), intent(out) *avalue*)

Definition at line 1828 of file mo_netcdf.f90.

References mo_ncread::check().

Here is the call graph for this function:

15.113.2.51 subroutine mo_netcdf::getglobalattributei64 (class(ncdataset), intent(in) *self*, character(∗), intent(in) *name*, integer(i8), intent(out) *avalue*)

Definition at line 1840 of file mo_netcdf.f90.

References mo_ncread::check().

Here is the call graph for this function:

15.113.2.52 subroutine mo_netcdf::getglobalattributei8 (class(ncdataset), intent(in) *self*, character(∗), intent(in) *name*, integer(i1), intent(out) *avalue*)

Definition at line 1804 of file mo_netcdf.f90.

References mo_ncread::check().

Here is the call graph for this function:

15.113.2.53 integer(i4) function mo_netcdf::getnodimensions (class(ncvariable), intent(in) *self*)

Definition at line 1650 of file mo_netcdf.f90.

References mo_ncread::check(), and mo_netcdf::ncdimension::getnodimensions().

Here is the call graph for this function:

15.113.2.54 integer(i4) function mo_netcdf::getnovariables (class(ncdataset), intent(in) *self*)

Definition at line 1410 of file mo_netcdf.f90.

References mo_ncread::check(), and mo_netcdf::ncdataset::getnovariables().

Here is the call graph for this function:

15.113.2.55 integer(i4) function, dimension(datarank) mo_netcdf::getreaddatashape (type(ncvariable), intent(in) *var,* integer(i4), intent(in) *datarank,* integer(i4), dimension(:), intent(in), optional *instart,* integer(i4), dimension(:), intent(in), optional *incnt,* integer(i4), dimension(:), intent(in), optional *instride*)

Definition at line 2953 of file mo_netcdf.f90.

Referenced by getdata1df32(), getdata1df64(), getdata1di16(), getdata1di32(), getdata1di64(), getdata1di8(), getdata2df32(), getdata2df64(), getdata2di16(), getdata2di32(), getdata2di64(), getdata2di8(), getdata3df32(), getdata3df64(), getdata3di16(), getdata3di32(), getdata3di64(), getdata3di8(), getdata4df32(), getdata4df64(), getdata4di16(), getdata4di32(), getdata4di64(), getdata4di8(), getdata5df32(), getdata5df64(), getdata5di16(), getdata5di32(), getdata5di64(), and getdata5di8().

Here is the caller graph for this function:

15.113.2.56 type(ncdimension) function mo_netcdf::getunlimiteddimension (class(ncdataset), intent(in) *self*)

Definition at line 1468 of file mo_netcdf.f90.

References mo_ncread::check(), and mo_netcdf::ncdataset::getunlimiteddimension().

Here is the call graph for this function:

15.113.2.57 subroutine mo_netcdf::getvariableattributechar (class(ncvariable), intent(in) *self*, character(∗), intent(in) *name*, character(∗), intent(out) *avalue*)

Definition at line 1939 of file mo_netcdf.f90.

References mo_ncread::check().

Here is the call graph for this function:

15.113.2.58 subroutine mo_netcdf::getvariableattributef32 (class(ncvariable), intent(in) *self*, character(∗), intent(in) *name*, real(sp), intent(out) *avalue*)

Definition at line 1999 of file mo_netcdf.f90.

References mo_ncread::check().

Here is the call graph for this function:

15.113.2.59 subroutine mo_netcdf::getvariableattributef64 (class(ncvariable), intent(in) *self*, character(∗), intent(in) *name*, real(dp), intent(out) *avalue*)

Definition at line 2011 of file mo_netcdf.f90.

References mo_ncread::check().

Here is the call graph for this function:

15.113.2.60 subroutine mo_netcdf::getvariableattributei16 (class(ncvariable), intent(in) *self*, character(∗), intent(in) *name*, integer(i2), intent(out) *avalue*)

Definition at line 1963 of file mo_netcdf.f90.

References mo_ncread::check().

Here is the call graph for this function:

15.113.2.61 subroutine mo_netcdf::getvariableattributei32 (class(ncvariable), intent(in) *self*, character(∗), intent(in) *name*, integer(i4), intent(out) *avalue*)

Definition at line 1975 of file mo_netcdf.f90.

References mo_ncread::check().

Here is the call graph for this function:

15.113.2.62 subroutine mo_netcdf::getvariableattributei64 (class(ncvariable), intent(in) *self*, character(*), intent(in) *name*, integer(i8), intent(out) *avalue*)

Definition at line 1987 of file mo_netcdf.f90.

References mo_ncread::check().

Here is the call graph for this function:

15.113.2.63 subroutine mo_netcdf::getvariableattributei8 (class(ncvariable), intent(in) *self*, character(*), intent(in) *name*, integer(i1), intent(out) *avalue*)

Definition at line 1951 of file mo_netcdf.f90.

References mo_ncread::check().

Here is the call graph for this function:

15.113.2.64 type(ncvariable) function mo_netcdf::getvariablebyname (class(ncdataset), intent(in) *self*, character(*), intent(in) *name*)

Definition at line 1630 of file mo_netcdf.f90.

References mo_ncread::check(), and mo_netcdf::ncdataset::getvariablebyname().

Here is the call graph for this function:

15.113.2.65 type(ncdimension) function, dimension(:), allocatable mo_netcdf::getvariabledimensions (class(ncvariable), intent(in) *self*)

Definition at line 1658 of file mo_netcdf.f90.

References mo_ncread::check().

Referenced by mo_netcdf::ncdimension::getdimensions().

Here is the call graph for this function:

Here is the caller graph for this function:

15.113.2.66 character(3) function mo_netcdf::getvariabledtype (class(ncvariable), intent(in) *self*)

Definition at line 1689 of file mo_netcdf.f90.

References mo_ncread::check(), and getdtypefrominteger().

Referenced by mo_netcdf::ncdimension::getdtype().

Here is the call graph for this function:

Here is the caller graph for this function:

15.113.2.67 subroutine mo_netcdf::getvariablefillvaluef32 (class(ncvariable), intent(in) *self*, real(sp), intent(out) *fvalue*)

Definition at line 2121 of file mo_netcdf.f90.

15.113.2.68 subroutine mo_netcdf::getvariablefillvaluef64 (class(ncvariable), intent(in) *self*, real(dp), intent(out) *fvalue*)

Definition at line 2132 of file mo_netcdf.f90.

15.113.2.69 subroutine mo_netcdf::getvariablefillvaluei16 (class(ncvariable), intent(in) *self*, integer(i2), intent(out) *fvalue*)

Definition at line 2088 of file mo_netcdf.f90.

15.113.2.70 subroutine mo_netcdf::getvariablefillvaluei32 (class(ncvariable), intent(in) *self*, integer(i4), intent(out) *fvalue*)

Definition at line 2099 of file mo_netcdf.f90.

15.113.2.71 subroutine mo_netcdf::getvariablefillvaluei64 (class(ncvariable), intent(in) *self*, integer(i8), intent(out) *fvalue*)

Definition at line 2110 of file mo_netcdf.f90.

15.113.2.72 subroutine mo_netcdf::getvariablefillvaluei8 (class(ncvariable), intent(in) *self*, integer(i1), intent(out) *fvalue*)

Definition at line 2077 of file mo_netcdf.f90.

15.113.2.73 integer(i4) function, dimension(:), allocatable mo_netcdf::getvariableids (class(ncdataset), intent(in) *self*)

Definition at line 1417 of file mo_netcdf.f90.

References mo_ncread::check(), and mo_netcdf::ncdataset::getvariableids().

Here is the call graph for this function:

15.113.2.74 character(len = 256) function mo_netcdf::getvariablename (class(ncvariable), intent(in) *self*)

Definition at line 1642 of file mo_netcdf.f90.

References mo_ncread::check().

Referenced by mo_netcdf::ncdimension::getname().

Here is the call graph for this function:

Here is the caller graph for this function:

15.113.2.75 type(ncvariable) function, dimension(:), allocatable mo_netcdf::getvariables (class(ncdataset), intent(in) *self*)

Definition at line 1426 of file mo_netcdf.f90.

References mo_netcdf::ncdataset::getvariables().

Here is the call graph for this function:

15.113.2.76 integer(i4) function, dimension(:), allocatable mo_netcdf::getvariableshape (class(ncvariable), intent(in) *self*)

Definition at line 1674 of file mo_netcdf.f90.

Referenced by mo_netcdf::ncdimension::getshape().

Here is the caller graph for this function:

15.113.2.77 logical function mo_netcdf::hasattribute (class(ncvariable), intent(in) *self*, character(∗), intent(in) *name*)

Definition at line 1719 of file mo_netcdf.f90.

References mo_netcdf::ncdimension::hasattribute().

Here is the call graph for this function:

15.113.2.78 logical function mo_netcdf::hasdimension (class(ncdataset), intent(in) *self*, character(∗), intent(in) *name*)

Definition at line 1528 of file mo_netcdf.f90.

References mo_netcdf::ncdataset::hasdimension().

Here is the call graph for this function:

15.113.2.79 logical function mo_netcdf::hasvariable (class(ncdataset), intent(in) *self*, character(∗), intent(in) *name*)

Definition at line 1519 of file mo_netcdf.f90.

References mo_netcdf::ncdataset::hasvariable().

Here is the call graph for this function:

15.113.2.80 subroutine mo_netcdf::initncdataset (class(ncdataset), intent(inout) *self*, character(∗), intent(in) *fname*, character(1), intent(in) *mode*)

Definition at line 1360 of file mo_netcdf.f90.

References mo_ncread::check().

Here is the call graph for this function:

15.113.2.81 subroutine mo_netcdf::initncdimension (class(ncdimension), intent(inout) *self*, integer(i4), intent(in) *id*, type(ncdataset), intent(in) *parent*)

Definition at line 1351 of file mo_netcdf.f90.

15.113.2.82 subroutine mo_netcdf::initncvariable (class(ncvariable), intent(inout) *self*, integer(i4), intent(in) *id*, type(ncdataset), intent(in) *parent*)

Definition at line 1342 of file mo_netcdf.f90.

15.113.2.83 logical function mo_netcdf::isdatasetunlimited (class(ncdataset), intent(in) *self*)

Definition at line 1458 of file mo_netcdf.f90.

References mo_ncread::check().

Referenced by mo_netcdf::ncdataset::isunlimited().

Here is the call graph for this function:

Here is the caller graph for this function:

15.113.2.84 logical function mo_netcdf::isunlimiteddimension (class(ncdimension), intent(in) *self*)

Definition at line 1490 of file mo_netcdf.f90.

15.113.2.85 logical function mo_netcdf::isunlimitedvariable (class(ncvariable), intent(in) *self*)

Definition at line 1699 of file mo_netcdf.f90.

Referenced by mo_netcdf::ncdimension::isunlimited().

Here is the caller graph for this function:

15.113.2.86 type(**ncdataset**) function mo_netcdf::newncdataset (character(*), intent(in) *fname,* character(1), intent(in) *mode*)

Definition at line 1397 of file mo_netcdf.f90.

Referenced by mo_netcdf::ncdataset::getvariable().

Here is the caller graph for this function:

15.113.2.87 type(**ncdimension**) function mo_netcdf::newncdimension (integer(i4), intent(in) *id,* type(**ncdataset**), intent(in) *parent*)

Definition at line 1390 of file mo_netcdf.f90.

15.113.2.88 type(**ncvariable**) function mo_netcdf::newncvariable (integer(i4), intent(in) *id,* type(**ncdataset**), intent(in) *parent*)

Definition at line 1383 of file mo_netcdf.f90.

15.113.2.89 subroutine mo_netcdf::setdata1df32 (class(**ncvariable**), intent(in) *self,* real(sp), dimension(:), intent(in) *values,* integer(i4), dimension(:), intent(in), optional *start,* integer(i4), dimension(:), intent(in), optional *cnt,* integer(i4), dimension(:), intent(in), optional *stride,* integer(i4), dimension(:), intent(in), optional *map*)

Definition at line 2368 of file mo_netcdf.f90.

References mo_ncread::check().

Here is the call graph for this function:

15.113.2.90 subroutine mo_netcdf::setdata1df64 (class(**ncvariable**), intent(in) *self,* real(dp), dimension(:), intent(in) *values,* integer(i4), dimension(:), intent(in), optional *start,* integer(i4), dimension(:), intent(in), optional *cnt,* integer(i4), dimension(:), intent(in), optional *stride,* integer(i4), dimension(:), intent(in), optional *map*)

Definition at line 2422 of file mo_netcdf.f90.

References mo_ncread::check().

Here is the call graph for this function:

15.113.2.91 subroutine mo_netcdf::setdata1di16 (class(**ncvariable**), intent(in) *self,* integer(i2), dimension(:), intent(in) *values,* integer(i4), dimension(:), intent(in), optional *start,* integer(i4), dimension(:), intent(in), optional *cnt,* integer(i4), dimension(:), intent(in), optional *stride,* integer(i4), dimension(:), intent(in), optional *map*)

Definition at line 2206 of file mo_netcdf.f90.

References mo_ncread::check().

Here is the call graph for this function:

15.113.2.92 subroutine mo_netcdf::setdata1di32 (class(ncvariable), intent(in) *self,* integer(i4), dimension(:), intent(in) *values,* integer(i4), dimension(:), intent(in), optional *start,* integer(i4), dimension(:), intent(in), optional *cnt,* integer(i4), dimension(:), intent(in), optional *stride,* integer(i4), dimension(:), intent(in), optional *map*)

Definition at line 2260 of file mo_netcdf.f90.

References mo_ncread::check().

Here is the call graph for this function:

15.113.2.93 subroutine mo_netcdf::setdata1di64 (class(ncvariable), intent(in) *self,* integer(i8), dimension(:), intent(in) *values,* integer(i4), dimension(:), intent(in), optional *start,* integer(i4), dimension(:), intent(in), optional *cnt,* integer(i4), dimension(:), intent(in), optional *stride,* integer(i4), dimension(:), intent(in), optional *map*)

Definition at line 2314 of file mo_netcdf.f90.

References mo_ncread::check().

Here is the call graph for this function:

15.113.2.94 subroutine mo_netcdf::setdata1di8 (class(ncvariable), intent(in) *self,* integer(i1), dimension(:), intent(in) *values,* integer(i4), dimension(:), intent(in), optional *start,* integer(i4), dimension(:), intent(in), optional *cnt,* integer(i4), dimension(:), intent(in), optional *stride,* integer(i4), dimension(:), intent(in), optional *map*)

Definition at line 2152 of file mo_netcdf.f90.

References mo_ncread::check().

Here is the call graph for this function:

15.113.2.95 subroutine mo_netcdf::setdata2df32 (class(ncvariable), intent(in) *self,* real(sp), dimension(:, :), intent(in) *values,* integer(i4), dimension(:), intent(in), optional *start,* integer(i4), dimension(:), intent(in), optional *cnt,* integer(i4), dimension(:), intent(in), optional *stride,* integer(i4), dimension(:), intent(in), optional *map*)

Definition at line 2377 of file mo_netcdf.f90.

References mo_ncread::check().

Here is the call graph for this function:

15.113.2.96 subroutine mo_netcdf::setdata2df64 (class(ncvariable), intent(in) *self,* real(dp), dimension(:, :), intent(in) *values,* integer(i4), dimension(:), intent(in), optional *start,* integer(i4), dimension(:), intent(in), optional *cnt,* integer(i4), dimension(:), intent(in), optional *stride,* integer(i4), dimension(:), intent(in), optional *map*)

Definition at line 2431 of file mo_netcdf.f90.

References mo_ncread::check().

Here is the call graph for this function:

15.113.2.97 subroutine mo_netcdf::setdata2di16 (class(**ncvariable**), intent(in) *self,* integer(i2), dimension(:, :), intent(in) *values,* integer(i4), dimension(:), intent(in), optional *start,* integer(i4), dimension(:), intent(in), optional *cnt,* integer(i4), dimension(:), intent(in), optional *stride,* integer(i4), dimension(:), intent(in), optional *map*)

Definition at line 2215 of file mo_netcdf.f90.

References mo_ncread::check().

Here is the call graph for this function:

15.113.2.98 subroutine mo_netcdf::setdata2di32 (class(**ncvariable**), intent(in) *self,* integer(i4), dimension(:, :), intent(in) *values,* integer(i4), dimension(:), intent(in), optional *start,* integer(i4), dimension(:), intent(in), optional *cnt,* integer(i4), dimension(:), intent(in), optional *stride,* integer(i4), dimension(:), intent(in), optional *map*)

Definition at line 2269 of file mo_netcdf.f90.

References mo_ncread::check().

Here is the call graph for this function:

15.113.2.99 subroutine mo_netcdf::setdata2di64 (class(**ncvariable**), intent(in) *self,* integer(i8), dimension(:, :), intent(in) *values,* integer(i4), dimension(:), intent(in), optional *start,* integer(i4), dimension(:), intent(in), optional *cnt,* integer(i4), dimension(:), intent(in), optional *stride,* integer(i4), dimension(:), intent(in), optional *map*)

Definition at line 2323 of file mo_netcdf.f90.

References mo_ncread::check().

Here is the call graph for this function:

15.113.2.100 subroutine mo_netcdf::setdata2di8 (class(**ncvariable**), intent(in) *self,* integer(i1), dimension(:, :), intent(in) *values,* integer(i4), dimension(:), intent(in), optional *start,* integer(i4), dimension(:), intent(in), optional *cnt,* integer(i4), dimension(:), intent(in), optional *stride,* integer(i4), dimension(:), intent(in), optional *map*)

Definition at line 2161 of file mo_netcdf.f90.

References mo_ncread::check().

Here is the call graph for this function:

15.113.2.101 subroutine mo_netcdf::setdata3df32 (class(**ncvariable**), intent(in) *self,* real(sp), dimension(:, :, :), intent(in) *values,* integer(i4), dimension(:), intent(in), optional *start,* integer(i4), dimension(:), intent(in), optional *cnt,* integer(i4), dimension(:), intent(in), optional *stride,* integer(i4), dimension(:), intent(in), optional *map*)

Definition at line 2386 of file mo_netcdf.f90.

References mo_ncread::check().

Here is the call graph for this function:

15.113.2.102 subroutine mo_netcdf::setdata3df64 (class(**ncvariable**), intent(in) *self,* real(dp), dimension(:, :, :), intent(in) *values,* integer(i4), dimension(:), intent(in), optional *start,* integer(i4), dimension(:), intent(in), optional *cnt,* integer(i4), dimension(:), intent(in), optional *stride,* integer(i4), dimension(:), intent(in), optional *map*)

Definition at line 2440 of file mo_netcdf.f90.

References mo_ncread::check().

Here is the call graph for this function:

15.113.2.103 subroutine mo_netcdf::setdata3di16 (class(**ncvariable**), intent(in) *self,* integer(i2), dimension(:, :, :), intent(in) *values,* integer(i4), dimension(:), intent(in), optional *start,* integer(i4), dimension(:), intent(in), optional *cnt,* integer(i4), dimension(:), intent(in), optional *stride,* integer(i4), dimension(:), intent(in), optional *map*)

Definition at line 2224 of file mo_netcdf.f90.

References mo_ncread::check().

Here is the call graph for this function:

15.113.2.104 subroutine mo_netcdf::setdata3di32 (class(**ncvariable**), intent(in) *self,* integer(i4), dimension(:, :, :), intent(in) *values,* integer(i4), dimension(:), intent(in), optional *start,* integer(i4), dimension(:), intent(in), optional *cnt,* integer(i4), dimension(:), intent(in), optional *stride,* integer(i4), dimension(:), intent(in), optional *map*)

Definition at line 2278 of file mo_netcdf.f90.

References mo_ncread::check().

Here is the call graph for this function:

15.113.2.105 subroutine mo_netcdf::setdata3di64 (class(**ncvariable**), intent(in) *self,* integer(i8), dimension(:, :, :), intent(in) *values,* integer(i4), dimension(:), intent(in), optional *start,* integer(i4), dimension(:), intent(in), optional *cnt,* integer(i4), dimension(:), intent(in), optional *stride,* integer(i4), dimension(:), intent(in), optional *map*)

Definition at line 2332 of file mo_netcdf.f90.

References mo_ncread::check().

Here is the call graph for this function:

15.113.2.106 subroutine mo_netcdf::setdata3di8 (class(**ncvariable**), intent(in) *self,* integer(i1), dimension(:, :, :), intent(in) *values,* integer(i4), dimension(:), intent(in), optional *start,* integer(i4), dimension(:), intent(in), optional *cnt,* integer(i4), dimension(:), intent(in), optional *stride,* integer(i4), dimension(:), intent(in), optional *map*)

Definition at line 2170 of file mo_netcdf.f90.

References mo_ncread::check().

Here is the call graph for this function:

15.113.2.107 subroutine mo_netcdf::setdata4df32 (class(**ncvariable**), intent(in) *self,* real(sp), dimension(:, :, :, :), intent(in) *values,* integer(i4), dimension(:), intent(in), optional *start,* integer(i4), dimension(:), intent(in), optional *cnt,* integer(i4), dimension(:), intent(in), optional *stride,* integer(i4), dimension(:), intent(in), optional *map*)

Definition at line 2395 of file mo_netcdf.f90.

References mo_ncread::check().

Here is the call graph for this function:

15.113.2.108 subroutine mo_netcdf::setdata4df64 (class(**ncvariable**), intent(in) *self,* real(dp), dimension(:, :, :, :), intent(in) *values,* integer(i4), dimension(:), intent(in), optional *start,* integer(i4), dimension(:), intent(in), optional *cnt,* integer(i4), dimension(:), intent(in), optional *stride,* integer(i4), dimension(:), intent(in), optional *map*)

Definition at line 2449 of file mo_netcdf.f90.

References mo_ncread::check().

Here is the call graph for this function:

15.113.2.109 subroutine mo_netcdf::setdata4di16 (class(**ncvariable**), intent(in) *self,* integer(i2), dimension(:, :, :, :), intent(in) *values,* integer(i4), dimension(:), intent(in), optional *start,* integer(i4), dimension(:), intent(in), optional *cnt,* integer(i4), dimension(:), intent(in), optional *stride,* integer(i4), dimension(:), intent(in), optional *map*)

Definition at line 2233 of file mo_netcdf.f90.

References mo_ncread::check().

Here is the call graph for this function:

15.113.2.110 subroutine mo_netcdf::setdata4di32 (class(**ncvariable**), intent(in) *self,* integer(i4), dimension(:, :, :, :), intent(in) *values,* integer(i4), dimension(:), intent(in), optional *start,* integer(i4), dimension(:), intent(in), optional *cnt,* integer(i4), dimension(:), intent(in), optional *stride,* integer(i4), dimension(:), intent(in), optional *map*)

Definition at line 2287 of file mo_netcdf.f90.

References mo_ncread::check().

Here is the call graph for this function:

15.113.2.111 subroutine mo_netcdf::setdata4di64 (class(**ncvariable**), intent(in) *self,* integer(i8), dimension(:, :, :, :), intent(in) *values,* integer(i4), dimension(:), intent(in), optional *start,* integer(i4), dimension(:), intent(in), optional *cnt,* integer(i4), dimension(:), intent(in), optional *stride,* integer(i4), dimension(:), intent(in), optional *map*)

Definition at line 2341 of file mo_netcdf.f90.

References mo_ncread::check().

Here is the call graph for this function:

15.113.2.112 subroutine mo_netcdf::setdata4di8 (class(**ncvariable**), intent(in) *self,* integer(i1), dimension(:, :, :, :), intent(in) *values,* integer(i4), dimension(:), intent(in), optional *start,* integer(i4), dimension(:), intent(in), optional *cnt,* integer(i4), dimension(:), intent(in), optional *stride,* integer(i4), dimension(:), intent(in), optional *map*)

Definition at line 2179 of file mo_netcdf.f90.

References mo_ncread::check().

Here is the call graph for this function:

15.113.2.113 subroutine mo_netcdf::setdata5df32 (class(**ncvariable**), intent(in) *self,* real(sp), dimension(:, :, :, :, :), intent(in) *values,* integer(i4), dimension(:), intent(in), optional *start,* integer(i4), dimension(:), intent(in), optional *cnt,* integer(i4), dimension(:), intent(in), optional *stride,* integer(i4), dimension(:), intent(in), optional *map*)

Definition at line 2404 of file mo_netcdf.f90.

References mo_ncread::check().

Here is the call graph for this function:

15.113.2.114 subroutine mo_netcdf::setdata5df64 (class(**ncvariable**), intent(in) *self,* real(dp), dimension(:, :, :, :, :), intent(in) *values,* integer(i4), dimension(:), intent(in), optional *start,* integer(i4), dimension(:), intent(in), optional *cnt,* integer(i4), dimension(:), intent(in), optional *stride,* integer(i4), dimension(:), intent(in), optional *map*)

Definition at line 2458 of file mo_netcdf.f90.

References mo_ncread::check().

Here is the call graph for this function:

15.113.2.115 subroutine mo_netcdf::setdata5di16 (class(**ncvariable**), intent(in) *self,* integer(i2), dimension(:, :, :, :, :), intent(in) *values,* integer(i4), dimension(:), intent(in), optional *start,* integer(i4), dimension(:), intent(in), optional *cnt,* integer(i4), dimension(:), intent(in), optional *stride,* integer(i4), dimension(:), intent(in), optional *map*)

Definition at line 2242 of file mo_netcdf.f90.

References mo_ncread::check().

Here is the call graph for this function:

15.113.2.116 subroutine mo_netcdf::setdata5di32 (class(**ncvariable**), intent(in) *self,* integer(i4), dimension(:, :, :, :, :), intent(in) *values,* integer(i4), dimension(:), intent(in), optional *start,* integer(i4), dimension(:), intent(in), optional *cnt,* integer(i4), dimension(:), intent(in), optional *stride,* integer(i4), dimension(:), intent(in), optional *map*)

Definition at line 2296 of file mo_netcdf.f90.

References mo_ncread::check().

Here is the call graph for this function:

15.113.2.117 subroutine mo_netcdf::setdata5di64 (class(**ncvariable**), intent(in) *self,* integer(i8), dimension(:, :, :, :, :), intent(in) *values,* integer(i4), dimension(:), intent(in), optional *start,* integer(i4), dimension(:), intent(in), optional *cnt,* integer(i4), dimension(:), intent(in), optional *stride,* integer(i4), dimension(:), intent(in), optional *map*)

Definition at line 2350 of file mo_netcdf.f90.

References mo_ncread::check().

Here is the call graph for this function:

15.113.2.118 subroutine mo_netcdf::setdata5di8 (class(**ncvariable**), intent(in) *self,* integer(i1), dimension(:, :, :, :, :), intent(in) *values,* integer(i4), dimension(:), intent(in), optional *start,* integer(i4), dimension(:), intent(in), optional *cnt,* integer(i4), dimension(:), intent(in), optional *stride,* integer(i4), dimension(:), intent(in), optional *map*)

Definition at line 2188 of file mo_netcdf.f90.

References mo_ncread::check().

Here is the call graph for this function:

15.113.2.119 subroutine mo_netcdf::setdatascalarf32 (class(**ncvariable**), intent(in) *self,* real(sp), intent(in) *values,* integer(i4), dimension(:), intent(in), optional *start*)

Definition at line 2359 of file mo_netcdf.f90.

References mo_ncread::check().

Here is the call graph for this function:

15.113.2.120 subroutine mo_netcdf::setdatascalarf64 (class(ncvariable), intent(in) *self*, real(dp), intent(in) *values*, integer(i4), dimension(:), intent(in), optional *start*)

Definition at line 2413 of file mo_netcdf.f90.

References mo_ncread::check().

Here is the call graph for this function:

15.113.2.121 subroutine mo_netcdf::setdatascalari16 (class(ncvariable), intent(in) *self*, integer(i2), intent(in) *values*, integer(i4), dimension(:), intent(in), optional *start*)

Definition at line 2197 of file mo_netcdf.f90.

References mo_ncread::check().

Here is the call graph for this function:

15.113.2.122 subroutine mo_netcdf::setdatascalari32 (class(ncvariable), intent(in) *self*, integer(i4), intent(in) *values*, integer(i4), dimension(:), intent(in), optional *start*)

Definition at line 2251 of file mo_netcdf.f90.

References mo_ncread::check().

Here is the call graph for this function:

15.113.2.123 subroutine mo_netcdf::setdatascalari64 (class(ncvariable), intent(in) *self*, integer(i8), intent(in) *values*, integer(i4), dimension(:), intent(in), optional *start*)

Definition at line 2305 of file mo_netcdf.f90.

References mo_ncread::check().

Here is the call graph for this function:

15.113.2.124 subroutine mo_netcdf::setdatascalari8 (class(ncvariable), intent(in) *self*, integer(i1), intent(in) *values*, integer(i4), dimension(:), intent(in), optional *start*)

Definition at line 2143 of file mo_netcdf.f90.

References mo_ncread::check().

Here is the call graph for this function:

15.113.2.125 type(ncdimension) function mo_netcdf::setdimension (class(ncdataset), intent(in) *self*, character(∗), intent(in) *name*, integer(i4), intent(in) *length*)

Definition at line 1500 of file mo_netcdf.f90.

References mo_ncread::check(), and mo_netcdf::ncdataset::setdimension().

Here is the call graph for this function:

15.113.2.126 subroutine mo_netcdf::setglobalattributechar (class(ncdataset), intent(in) *self*, character(∗), intent(in) *name*, character(∗), intent(in) *data*)

Definition at line 1729 of file mo_netcdf.f90.

References mo_ncread::check().

Here is the call graph for this function:

15.113.2.127 subroutine mo_netcdf::setglobalattributef32 (class(ncdataset), intent(in) *self*, character(∗), intent(in) *name*, real(sp), intent(in) *data*)

Definition at line 1774 of file mo_netcdf.f90.

References mo_ncread::check().

Here is the call graph for this function:

15.113.2.128 subroutine mo_netcdf::setglobalattributef64 (class(ncdataset), intent(in) *self*, character(*), intent(in) *name*, real(dp), intent(in) *data*)

Definition at line 1783 of file mo_netcdf.f90.

References mo_ncread::check().

Here is the call graph for this function:

15.113.2.129 subroutine mo_netcdf::setglobalattributei16 (class(ncdataset), intent(in) *self*, character(*), intent(in) *name*, integer(i2), intent(in) *data*)

Definition at line 1747 of file mo_netcdf.f90.

References mo_ncread::check().

Here is the call graph for this function:

15.113.2.130 subroutine mo_netcdf::setglobalattributei32 (class(ncdataset), intent(in) *self*, character(*), intent(in) *name*, integer(i4), intent(in) *data*)

Definition at line 1756 of file mo_netcdf.f90.

References mo_ncread::check().

Here is the call graph for this function:

15.113.2.131 subroutine mo_netcdf::setglobalattributei64 (class(ncdataset), intent(in) *self*, character(∗), intent(in) *name*, integer(i8), intent(in) *data*)

Definition at line 1765 of file mo_netcdf.f90.

References mo_ncread::check().

Here is the call graph for this function:

15.113.2.132 subroutine mo_netcdf::setglobalattributei8 (class(ncdataset), intent(in) *self*, character(∗), intent(in) *name*, integer(i1), intent(in) *data*)

Definition at line 1738 of file mo_netcdf.f90.

References mo_ncread::check().

Here is the call graph for this function:

15.113.2.133 subroutine mo_netcdf::setvariableattributechar (class(ncvariable), intent(in) *self*, character(∗), intent(in) *name*, character(∗), intent(in) *data*)

Definition at line 1876 of file mo_netcdf.f90.

References mo_ncread::check().

Here is the call graph for this function:

15.113.2.134 subroutine mo_netcdf::setvariableattributef32 (class(ncvariable), intent(in) *self*, character(∗), intent(in) *name*, real(sp), intent(in) *data*)

Definition at line 1921 of file mo_netcdf.f90.

References mo_ncread::check().

Here is the call graph for this function:

15.113.2.135 subroutine mo_netcdf::setvariableattributef64 (class(ncvariable), intent(in) *self*, character(∗), intent(in) *name*, real(dp), intent(in) *data*)

Definition at line 1930 of file mo_netcdf.f90.

References mo_ncread::check().

Here is the call graph for this function:

15.113.2.136 subroutine mo_netcdf::setvariableattributei16 (class(ncvariable), intent(in) *self*, character(∗), intent(in) *name*, integer(i2), intent(in) *data*)

Definition at line 1894 of file mo_netcdf.f90.

References mo_ncread::check().

Here is the call graph for this function:

15.113.2.137 subroutine mo_netcdf::setvariableattributei32 (class(ncvariable), intent(in) *self*, character(∗), intent(in) *name*, integer(i4), intent(in) *data*)

Definition at line 1903 of file mo_netcdf.f90.

References mo_ncread::check().

Here is the call graph for this function:

15.113.2.138 subroutine mo_netcdf::setvariableattributei64 (class(ncvariable), intent(in) *self*, character(∗), intent(in) *name*, integer(i8), intent(in) *data*)

Definition at line 1912 of file mo_netcdf.f90.

References mo_ncread::check().

Here is the call graph for this function:

15.113.2.139 subroutine mo_netcdf::setvariableattributei8 (class(ncvariable), intent(in) *self*, character(∗), intent(in) *name*, integer(i1), intent(in) *data*)

Definition at line 1885 of file mo_netcdf.f90.

References mo_ncread::check().

Here is the call graph for this function:

15.113.2.140 subroutine mo_netcdf::setvariablefillvaluef32 (class(ncvariable), intent(in) *self,* real(sp), intent(in) *fvalue*)

Definition at line 2059 of file mo_netcdf.f90.

15.113.2.141 subroutine mo_netcdf::setvariablefillvaluef64 (class(ncvariable), intent(in) *self,* real(dp), intent(in) *fvalue*)

Definition at line 2068 of file mo_netcdf.f90.

15.113.2.142 subroutine mo_netcdf::setvariablefillvaluei16 (class(ncvariable), intent(in) *self,* integer(i2), intent(in) *fvalue*)

Definition at line 2032 of file mo_netcdf.f90.

15.113.2.143 subroutine mo_netcdf::setvariablefillvaluei32 (class(ncvariable), intent(in) *self,* integer(i4), intent(in) *fvalue*)

Definition at line 2041 of file mo_netcdf.f90.

15.113.2.144 subroutine mo_netcdf::setvariablefillvaluei64 (class(ncvariable), intent(in) *self,* integer(i8), intent(in) *fvalue*)

Definition at line 2050 of file mo_netcdf.f90.

15.113.2.145 subroutine mo_netcdf::setvariablefillvaluei8 (class(ncvariable), intent(in) *self,* integer(i1), intent(in) *fvalue*)

Definition at line 2023 of file mo_netcdf.f90.

15.113.2.146 type(ncvariable) function mo_netcdf::setvariablewithids (class(ncdataset), intent(in) *self,* character(∗), intent(in) *name,* character(3), intent(in) *dtype,* integer(i4), dimension(:), intent(in) *dimensions,* logical, intent(in), optional *contiguous,* integer(i4), dimension(:), intent(in), optional *chunksizes,* integer(i4), intent(in), optional *deflate_level,* logical, intent(in), optional *shuffle,* logical, intent(in), optional *fletcher32,* integer(i4), intent(in), optional *endianness,* integer(i4), intent(in), optional *cache_size,* integer(i4), intent(in), optional *cache_nelems,* integer(i4), intent(in), optional *cache_preemption*)

Definition at line 1538 of file mo_netcdf.f90.

References mo_ncread::check(), getdtypefromstring(), and mo_netcdf::ncdataset::setvariablewithids().

Here is the call graph for this function:

15.113.2.147 type(**ncvariable**) function mo_netcdf::setvariablewithnames (class(**ncdataset**), intent(in) *self,* character(∗), intent(in) *name,* character(3), intent(in) *dtype,* character(∗), dimension(:), intent(in) *dimensions,* logical, intent(in), optional *contiguous,* integer(i4), dimension(:), intent(in), optional *chunksizes,* integer(i4), intent(in), optional *deflate_level,* logical, intent(in), optional *shuffle,* logical, intent(in), optional *fletcher32,* integer(i4), intent(in), optional *endianness,* integer(i4), intent(in), optional *cache_size,* integer(i4), intent(in), optional *cache_nelems,* integer(i4), intent(in), optional *cache_preemption*)

Definition at line 1558 of file mo_netcdf.f90.

References mo_netcdf::ncdataset::setvariablewithids(), and mo_netcdf::ncdataset::setvariablewithnames().

Here is the call graph for this function:

15.113.2.148 type(**ncvariable**) function mo_netcdf::setvariablewithtypes (class(**ncdataset**), intent(in) *self,* character(∗), intent(in) *name,* character(3), intent(in) *dtype,* type(**ncdimension**), dimension(:), intent(in) *dimensions,* logical, intent(in), optional *contiguous,* integer(i4), dimension(:), intent(in), optional *chunksizes,* integer(i4), intent(in), optional *deflate_level,* logical, intent(in), optional *shuffle,* logical, intent(in), optional *fletcher32,* integer(i4), intent(in), optional *endianness,* integer(i4), intent(in), optional *cache_size,* integer(i4), intent(in), optional *cache_nelems,* integer(i4), intent(in), optional *cache_preemption*)

Definition at line 1583 of file mo_netcdf.f90.

References mo_netcdf::ncdataset::setvariablewithids(), and mo_netcdf::ncdataset::setvariablewithtypes().

Here is the call graph for this function:

The documentation for this module was generated from the following file:

- mo_netcdf.f90

15.114 mo_neutrons Module Reference

Models to predict neutron intensities above soils.

Public Member Functions

- 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.
- real(dp) function intgrandfast (c, phi)

  TODO: add description.

Private Member Functions

- 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.

15.114.1 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.

Definition at line 16 of file mo_neutrons.f90.

15.114.2 Member Function/Subroutine Documentation

15.114.2.1 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:

15.114.2.2 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:

15.114.2.3 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:

15.114.2.4 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 lookupintegral().

Referenced by mo_mhm::mhm().

Here is the call graph for this function:

Here is the caller graph for this function:

15.114.2.5 subroutine, public mo_neutrons::desiletsn0 (real(dp), dimension(:), intent(in) *SoilMoisture,* real(dp), dimension(:), intent(in) *Horizons,* real(dp), intent(in) *N0,* real(dp), intent(inout) *neutrons*)

Calculate neutrons from soil moisture in the first layer.

Using the N0-relation derived by Desilets, neutron counts above the ground (one value per cell in mHM) can be derived by a semi-empirical, semi-physical relation. The result depends on N0, the neutron counts for 0% soil mositure. This variable is site-specific and is a global parameter in mHM. N0 formula based on Desilets et al. 2010 Horizons(1) must not be zero. N0=1500cph, SoilMoisture(1,1)=700mm, Horizons(1)=200mm 1500∗(0.372+0.0808/(70mm/200mm + 0.115)) DesiletsN0 = 819cph Desilets, D., M. Zreda, and T. P. A. Ferre (2010), Nature's neutron probe: Land surface hydrology at an elusive scale with cosmic rays, WRR, 46, W11505, doi:10.1029/2009W↩R008726.

Parameters

in	*real(dp), dimension(:) :: SoilMoisture*	Soil Moisture
in	*real(dp), dimension(:) :: Horizons*	Horizon depths
in	*real(dp) :: N0*	dry neutron counts
in,out	*real(dp) :: neutrons*	Neutron counts

Authors

Martin Schroen

Date

Mar 2015

Definition at line 78 of file mo_neutrons.f90.

Referenced by mo_mhm::mhm().

Here is the caller graph for this function:

15.114.2.6 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:

15.114.2.7 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:

15.114.2.8 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.

15.114.2.9 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:

The documentation for this module was generated from the following file:

- mo_neutrons.f90

15.115 mo_nml Module Reference

Deal with namelist files.

Collaboration diagram for mo_nml:

Public Member Functions

- subroutine, public open_nml (file, unit, quiet)

  Open a namelist file.
- subroutine, public close_nml (unit)

  Close a namelist file.
- subroutine, public position_nml (name, unit, status, first)

  Position a namlist file.

Public Attributes

- integer(i4), parameter, public positioned = 0

  Information: file pointer set to namelist group.
- integer(i4), parameter, public missing = 1

  Error: namelist group is missing.

- integer(i4), parameter, public length_error = 2

  Error: namelist group name too long.
- integer(i4), parameter, public read_error = 3

  Error occured during read of namelist file.
- integer, save, public nunitnml = -1

15.115.1 Detailed Description

Deal with namelist files.

This module provides routines to open, close and position namelist files.

Authors

Matthias Cuntz

Date

Jan 2011

Definition at line 10 of file mo_nml.f90.

15.115.2 Member Function/Subroutine Documentation

15.115.2.1 subroutine, public mo_nml::close_nml (integer, intent(in), optional *unit*)

Close a namelist file.

Parameters

in	*integer, optional :: unit*	namelist unit

Author

Matthias Cuntz - modified from Echam5, (C) MPI-MET, Hamburg, Germany

Date

Dec 2011

Definition at line 195 of file mo_nml.f90.

References mo_finish::finish().

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

Here is the call graph for this function:

Here is the caller graph for this function:

15.115.2.2 subroutine, public mo_nml::open_nml (character(len = ∗), intent(in) *file,* integer, intent(in) *unit,* logical, intent(in), optional *quiet*)

Open a namelist file.

Parameters

in	*character(len=∗) :: file*	namelist filename
in	*integer :: unit*	namelist unit
in	*logical, optional :: quiet*	Be verbose or not (default: .true.) .true.: no messages .false.: write out messages

Author

Matthias Cuntz - modified from Echam5, (C) MPI-MET, Hamburg, Germany

Date

Dec 2011

Definition at line 124 of file mo_nml.f90.

References mo_finish::finish(), and mo_message::message().

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

Here is the call graph for this function:

Here is the caller graph for this function:

15.115.2.3 subroutine, public mo_nml::position_nml (character(len = ∗), intent(in) *name,* integer, intent(in), optional *unit,* integer(i4), intent(out), optional *status,* logical, intent(in), optional *first*)

Position a namlist file.

Position namelist file pointer for reading a new namelist next.
It positions the namelist file at the correct place for reading
namelist /name/ (case independent).

Parameters

in	*character(len=∗) :: name*	namelist name (case independent)
in	*integer, optional :: unit*	namelist unit (default: nunitnml)
in	*logical, optional :: first*	start search at beginning, i.e. rewind the namelist first (default: .true.) .true.: rewind .false.: continue search from current file pointer
out	*integer(i4), optional :: status*	Set on output to either of POSITIONED (0) - correct MISSING (1) - name not found LENGTH_ERROR (2) - namelist length longer then 256 characters READ_ERROR (3) - error while reading namelist file

Author

Matthias Cuntz - modified from Echam5, (C) MPI-MET, Hamburg, Germany

Date

Dec 2011

Definition at line 272 of file mo_nml.f90.

References mo_finish::finish(), and mo_string_utils::tolower().

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

Here is the call graph for this function:

Here is the caller graph for this function:

15.115.3 Member Data Documentation

15.115.3.1 integer(i4), parameter, public mo_nml::length_error = 2

Error: namelist group name too long.

Definition at line 65 of file mo_nml.f90.

15.115.3.2 integer(i4), parameter, public mo_nml::missing = 1

Error: namelist group is missing.

Definition at line 63 of file mo_nml.f90.

15.115.3.3 integer, save, public mo_nml::nunitnml = -1

Definition at line 70 of file mo_nml.f90.

15.115.3.4 integer(i4), parameter, public mo_nml::positioned = 0

Information: file pointer set to namelist group.

Definition at line 61 of file mo_nml.f90.

15.115.3.5 integer(i4), parameter, public mo_nml::read_error = 3

Error occured during read of namelist file.

Definition at line 67 of file mo_nml.f90.

The documentation for this module was generated from the following file:

- mo_nml.f90

15.116 mo_objective_function Module Reference

Objective Functions for Optimization of mHM.

Public Member Functions

- real(dp) function, public objective (parameterset, eval, arg1, arg2, arg3)

  Wrapper for objective functions.
- real(dp) function objective_sm_kge_catchment_avg (parameterset, eval)

  Objective function for soil moisture.
- real(dp) function objective_sm_corr (parameterset, eval)

  Objective function for soil moisture.
- real(dp) function objective_sm_pd (parameterset, eval)

  Objective function for soil moisture.
- real(dp) function objective_sm_sse_standard_score (parameterset, eval)

  Objective function for soil moisture.
- real(dp) function objective_kge_q_rmse_tws (parameterset, eval)

  Objective function of KGE for runoff and RMSE for basin_avg TWS (standarized scores)
- real(dp) function objective_neutrons_kge_catchment_avg (parameterset, eval)

  Objective function for neutrons.
- real(dp) function objective_et_kge_catchment_avg (parameterset, eval)

  Objective function for evpotranspirstion (et).
- real(dp) function objective_kge_q_sm_corr (parameterset, eval)

  Objective function of KGE for runoff and correlation for SM.
- real(dp) function objective_kge_q_et (parameterset, eval)

  Objective function of KGE for runoff and KGE for ET.
- real(dp) function objective_kge_q_rmse_et (parameterset, eval)

  Objective function of KGE for runoff and RMSE for basin_avg ET (standarized scores)
- subroutine extract_basin_avg_tws (basinId, tws, tws_sim, tws_obs, tws_obs_mask)

  extracts basin average tws data from global variables

15.116.1 Detailed Description

Objective Functions for Optimization of mHM.

This module provides a wrapper for several objective functions used to optimize mHM against various variables. If the objective is only regarding runoff move it to mRM/mo_mrm_objective_function_runoff.f90. If it contains besides runoff another variable like TWS implement it here. All the objective functions are supposed to be minimized! (10) SO: SM: 1.0 - KGE of catchment average soilmoisture (11) SO: SM: 1.0 - Pattern dissimilarity (PD) of spatially distributed soil moisture (12) SO: SM: Sum of squared errors (SSE) of spatially distributed standard score (normalization) of soil moisture (13) SO: SM: 1.0 - average temporal correlation of spatially distributed soil moisture (15) SO: Q + TWS: [1.0-KGE(Q)]∗RMSE(basin_avg_TWS) - objective function using Q and basin average (standard score) TWS (17) SO: N: 1.0 - KGE of spatio-temporal neutron data, catchment-average (27) SO: ET: 1.0 - KGE of catchment average evapotranspiration

Authors

Juliane Mai

Date

Dec 2012

Definition at line 29 of file mo_objective_function.f90.

15.116.2 Member Function/Subroutine Documentation

15.116.2.1 subroutine mo_objective_function::extract_basin_avg_tws (integer(i4), intent(in) *basinId,* real(dp), dimension(:, :), intent(in) *tws,* real(dp), dimension(:), intent(out), allocatable *tws_sim,* real(dp), dimension(:), intent(out), allocatable *tws_obs,* logical, dimension(:), intent(out), allocatable *tws_obs_mask*)

extracts basin average tws data from global variables

extracts simulated and measured basin average tws from global variables, such that they overlay exactly. For measured tws, only the tws during the evaluation period are cut, not succeeding nodata values. For simulated tws, warming days as well as succeeding nodata values are neglected. see use in this module above

Parameters

in	*integer(i4) :: basinId*	current basin Id
in	*real(dp), dimension(:, :) :: tws*	simulated basin average tws
out	*real(dp), dimension(:) :: tws_sim*	aggregated simulated
out	*real(dp), dimension(:) :: tws_obs*	extracted measured
out	*logical, dimension(:) :: tws_obs_mask*	mask of no data values

Authors

Stephan Thober

Date

Oct 2015

Definition at line 1968 of file mo_objective_function.f90.

References mo_message::message().

Referenced by objective_kge_q_rmse_tws().

Here is the call graph for this function:

Here is the caller graph for this function:

15.116.2.2 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 85 of file mo_objective_function.f90.

References mo_message::message(), objective_et_kge_catchment_avg(), objective_kge_q_et(), objective_kge_q_rmse_et(), objective_kge_q_rmse_tws(), objective_kge_q_sm_corr(), objective_neutrons_kge_catchment_avg(), objective_sm_corr(), objective_sm_kge_catchment_avg(), objective_sm_pd(), and objective_sm_sse_standard_score().

Referenced by mhm_driver().

Here is the call graph for this function:

Here is the caller graph for this function:

15.116.2.3 real(dp) function mo_objective_function::objective_et_kge_catchment_avg (real(dp), dimension(:), intent(in) *parameterset,* procedure(eval_interface), intent(in), pointer *eval*)

Objective function for evpotranspirstion (et).

The objective function only depends on a parameter vector. The model will be called with that parameter vector and the model output is subsequently compared to observed data. Therefore, the Kling-Gupta model efficiency KGE of the catchment average evapotranspiration (et) is calculated

$$KGE = 1.0 - \sqrt{((1-r)^2 + (1-\alpha)^2 + (1-\beta)^2)}$$

where r = Pearson product-moment correlation coefficient α = ratio of simulated mean to observed mean SM β = ratio of similated standard deviation to observed standard deviation is calculated and the objective function for a given basin i is

$$\phi_i = 1.0 - KGE_i$$

ϕ_i is the objective since we always apply minimization methods. The minimal value of ϕ_i is 0 for the optimal KGE of 1.0. Finally, the overall objective function value OF is estimated based on the power-6 norm to combine the ϕ_i from all basins N.

$$OF = \sqrt[6]{\sum((1.0 - KGE_i)/N)^6}.$$

The observed data L1_et, L1_et_mask are global in this module.

Parameters

in	*real(dp), dimension(:) :: parameterset*	
in	*procedure(eval↩ _interface) :: eval*	

Returns

real(dp) :: objective_et_kge_catchment_avg — objective function value (which will be e.g. minimized by an optimization routine)

Authors

Johannes Brenner

Date

Feb 2017

Definition at line 1175 of file mo_objective_function.f90.

References mo_message::message().

Referenced by objective().

Here is the call graph for this function:

Here is the caller graph for this function:

15.116.2.4 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

Definition at line 1488 of file mo_objective_function.f90.

References mo_mrm_objective_function_runoff::extract_runoff(), and mo_message::message().

Referenced by objective().

Here is the call graph for this function:

Here is the caller graph for this function:

15.116.2.5 real(dp) function mo_objective_function::objective_kge_q_rmse_et (real(dp), dimension(:), intent(in) *parameterset*, procedure(eval_interface), intent(in), pointer *eval*)

Objective function of KGE for runoff and RMSE for basin_avg ET (standarized scores)

Objective function of KGE for runoff and RMSE for basin_avg ET (standarized scores)

Parameters

in	*real(dp), dimension(:) :: parameterset*	
in	*procedure(eval↩ _interface) :: eval*	

Returns

real(dp) :: objective_kge_q_rmse_et — objective function value (which will be e.g. minimized by an optimization routine like DDS)

Authors

Johannes Brenner

Date

July 2017

Definition at line 1669 of file mo_objective_function.f90.

References mo_julian::caldat(), mo_mrm_objective_function_runoff::extract_runoff(), and mo_message↩::message().

Referenced by objective().

Here is the call graph for this function:

Here is the caller graph for this function:

15.116.2.6 real(dp) function mo_objective_function::objective_kge_q_rmse_tws (real(dp), dimension(:), intent(in) *parameterset,* procedure(eval_interface), intent(in), pointer *eval*)

Objective function of KGE for runoff and RMSE for basin_avg TWS (standarized scores)

Objective function of KGE for runoff and RMSE for basin_avg TWS (standarized scores)

Parameters

in	*real(dp), dimension(:) :: parameterset*	
in	*procedure(eval↩ _interface) :: eval*	

Returns

real(dp) :: objective_kge_q_rmse_tws — objective function value (which will be e.g. minimized by an optimization routine like DDS)

Authors

Oldrich Rakovec, Rohini Kumar

Date

Oct. 2015

Definition at line 772 of file mo_objective_function.f90.

References mo_julian::caldat(), extract_basin_avg_tws(), mo_mrm_objective_function_runoff::extract_runoff(), and mo_message::message().

Referenced by objective().

Here is the call graph for this function:

Here is the caller graph for this function:

15.116.2.7 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 1303 of file mo_objective_function.f90.

References mo_mrm_objective_function_runoff::extract_runoff(), mo_message::message(), and mo_mrm_↩ objective_function_runoff::objective_kge().

Referenced by objective().

Here is the call graph for this function:

Here is the caller graph for this function:

15.116.2.8 real(dp) function mo_objective_function::objective_neutrons_kge_catchment_avg (real(dp), dimension(:), intent(in) *parameterset,* procedure(eval_interface), intent(in), pointer *eval*)

Objective function for neutrons.

The objective function only depends on a parameter vector. The model will be called with that parameter vector and the model output is subsequently compared to observed data. Therefore, the Kling-Gupta model efficiency KGE of the catchment average neutrons (N) is calculated

$$KGE = 1.0 - \sqrt{((1-r)^2 + (1-\alpha)^2 + (1-\beta)^2)}$$

where r = Pearson product-moment CORRELATION coefficient α = ratio of simulated mean to observed mean SM β = ratio of similated standard deviation to observed standard deviation is calculated and the objective function for a given basin i is

$$\phi_i = 1.0 - KGE_i$$

ϕ_i is the objective since we always apply minimization methods. The minimal value of ϕ_i is 0 for the optimal KGE of 1.0. Finally, the overall objective function value OF is estimated based on the power-6 norm to combine the ϕ_i from all basins N.

$$OF = \sqrt[6]{\sum((1.0 - KGE_i)/N)^6}.$$

The observed data L1_neutronsdata, L1_neutronsdata_mask are global in this module.

Parameters

in	*real(dp), dimension(:) :: parameterset*	
in	*procedure(eval↩ _interface) :: eval*	

Returns

real(dp) :: objective_neutrons_kge_catchment_avg — objective function value (which will be e.g. minimized by an optimization routine)

Authors

Martin Schroen

Date

Jun 2015

Definition at line 1031 of file mo_objective_function.f90.

References mo_message::message().

Referenced by objective().

Here is the call graph for this function:

Here is the caller graph for this function:

15.116.2.9 real(dp) function mo_objective_function::objective_sm_corr (real(dp), dimension(:), intent(in) *parameterset,* procedure(eval_interface), intent(in), pointer *eval*)

Objective function for soil moisture.

The objective function only depends on a parameter vector. The model will be called with that parameter vector and the model output is subsequently compared to observed data. Therefore the Pearson correlation between observed and modeled soil moisture on each grid cell j is compared

$$r_j = r^2(SM^j_{obs}, SM^j_{sim})$$

where r^2 = Pearson correlation coefficient, SM_{obs} = observed soil moisture, SM_{sim} = simulated soil moisture. The observed data SM_{obs} are global in this module. The the correlation is spatially averaged as

$$\phi_i = \frac{1}{K} \cdot \sum_{j=1}^{K} r_j$$

where K denotes the number of valid cells in the study domain. Finally, the overall objective function value OF is estimated based on the power-6 norm to combine the ϕ_i from all basins N.

$$OF = \sqrt[6]{\sum((1.0 - \phi_i)/N)^6}.$$

The observed data L1_sm, L1_sm_mask are global in this module.

Parameters

in	*real(dp), dimension(:) :: parameterset*	
in	*procedure(eval↩ _interface) :: eval*	

Returns

real(dp) :: objective_sm_corr — objective function value (which will be e.g. minimized by an optimization routine like DDS)

Authors

Matthias Zink

Date

March 2015

Definition at line 365 of file mo_objective_function.f90.

References mo_message::message().

Referenced by objective().

Here is the call graph for this function:

Here is the caller graph for this function:

15.116.2.10 real(dp) function mo_objective_function::objective_sm_kge_catchment_avg (real(dp), dimension(:), intent(in) *parameterset,* procedure(eval_interface), intent(in), pointer *eval*)

Objective function for soil moisture.

The objective function only depends on a parameter vector. The model will be called with that parameter vector and the model output is subsequently compared to observed data. Therefore, the Kling-Gupta model efficiency KGE of the catchment average soil mloisture (SM) is calculated

$$KGE = 1.0 - \sqrt{((1-r)^2 + (1-\alpha)^2 + (1-\beta)^2)}$$

where r = Pearson product-moment correlation coefficient α = ratio of simulated mean to observed mean SM β = ratio of similated standard deviation to observed standard deviation is calculated and the objective function for a given basin i is

$$\phi_i = 1.0 - KGE_i$$

ϕ_i is the objective since we always apply minimization methods. The minimal value of ϕ_i is 0 for the optimal KGE of 1.0. Finally, the overall objective function value OF is estimated based on the power-6 norm to combine the ϕ_i from all basins N.

$$OF = \sqrt[6]{\sum((1.0 - KGE_i)/N)^6}.$$

The observed data L1_sm, L1_sm_mask are global in this module.

Parameters

in	*real(dp), dimension(:) :: parameterset*	
in	*procedure(eval↩ _interface) :: eval*	

Returns

real(dp) :: objective_sm_kge_catchment_avg — objective function value (which will be e.g. minimized by an optimization routine like DDS)

Authors

Matthias Zink

Date

May 2015

Definition at line 203 of file mo_objective_function.f90.

References mo_message::message().

Referenced by objective().

Here is the call graph for this function:

Here is the caller graph for this function:

15.116.2.11 real(dp) function mo_objective_function::objective_sm_pd (real(dp), dimension(:), intent(in) *parameterset*, procedure(eval_interface), intent(in), pointer *eval*)

Objective function for soil moisture.

The objective function only depends on a parameter vector. The model will be called with that parameter vector and the model output is subsequently compared to observed data. Therefore the Pattern Dissimilarity (PD) of observed and modeled soil moisture fields is calculated - aim: matching spatial patters

$$E(t) = PD\left(SM_{obs}(t), SM_{sim}(t)\right)$$

where PD = pattern dissimilarity function, SM_{obs} = observed soil moisture, SM_{sim} = simulated soil moisture. $E(t)$ = pattern dissimilarity at timestep t. The the pattern dissimilaity (E) is spatially averaged as

$$\phi_i = \frac{1}{T} \cdot \sum_{t=1}^{T} E_t$$

where T denotes the number of time steps. Finally, the overall objective function value OF is estimated based on the power-6 norm to combine the ϕ_i from all basins N.

$$OF = \sqrt[6]{\sum((1.0-\phi_i)/N)^6}.$$

The observed data L1_sm, L1_sm_mask are global in this module. The observed data L1_sm, L1_sm_mask are global in this module.

Parameters

in	*real(dp), dimension(:) :: parameterset*	
in	*procedure(eval↩ _interface) :: eval*	

Returns

real(dp) :: objecive_sm_pd — objective function value (which will be e.g. minimized by an optimization routine like DDS)

Authors

Matthias Zink

Date

May 2015

Definition at line 500 of file mo_objective_function.f90.

References mo_message::message().

Referenced by objective().

Here is the call graph for this function:

Here is the caller graph for this function:

15.116.2.12 real(dp) function mo_objective_function::objective_sm_sse_standard_score (real(dp), dimension(:), intent(in) *parameterset*, procedure(eval_interface), intent(in), pointer *eval*)

Objective function for soil moisture.

The objective function only depends on a parameter vector. The model will be called with that parameter vector and the model output is subsequently compared to observed data. Therefore the sum of squared errors (SSE) of the standard score of observed and modeled soil moisture is calculated. The standard score or normalization (anomaly) make the objctive function bias insensitive and basically the dynamics of the soil moisture is tried to capture by this obejective function.

$$phi_i = \sum_{j=1}^{K} \{standard_score(SM_{obs}(j)) - standard_score(SM_{sim}(j))\}^2$$

where $standard_score$ = standard score function, SM_{obs} = observed soil moisture, SM_{sim} = simulated soil moisture. K = valid elements in study domain. Finally, the overall objective function value OF is estimated based on the power-6 norm to combine the ϕ_i from all basins N.

$$OF = \sqrt[6]{\sum(\phi_i/N)^6}.$$

The observed data L1_sm, L1_sm_mask are global in this module.

Parameters

in	*real(dp), dimension(:) :: parameterset*	
in	*procedure(eval↩ _interface) :: eval*	

Returns

real(dp) :: objective_sm_sse_standard_score — objective function value (which will be e.g. minimized by an optimization routine like DDS)

Authors

Matthias Zink

Date

March 2015

Definition at line 653 of file mo_objective_function.f90.

References mo_message::message().

Referenced by objective().

Here is the call graph for this function:

Here is the caller graph for this function:

The documentation for this module was generated from the following file:

- mo_objective_function.f90

15.117 mo_optimization_utils Module Reference

Data Types

- interface eval_interface
- interface objective_interface

15.117.1 Detailed Description

Definition at line 1 of file mo_optimization_utils.f90.

The documentation for this module was generated from the following file:

- mo_optimization_utils.f90

15.118 mo_orderpack Module Reference

Sort and ranking routines.

Collaboration diagram for mo_orderpack:

Data Types

- interface ctrper
- interface fndnth
- interface indmed
- interface indnth
- interface inspar
- interface inssor
- interface mrgref
- interface mrgrnk
- interface mulcnt
- interface nearless
- interface omedian
- interface rapknr
- interface refpar
- interface refsor
- interface rinpar
- interface rnkpar
- interface sort

  Unconditional ranking.
- interface sort_index
- interface uniinv
- interface unipar
- interface unirnk
- interface unista
- interface valmed
- interface valnth

Private Member Functions

- integer(i4) function, dimension(size(arr)) sort_index_dp (arr)
- integer(i4) function, dimension(size(arr)) sort_index_sp (arr)

- integer(i4) function, dimension(size(arr)) sort_index_i4 (arr)
- subroutine, private d_ctrper (XDONT, PCLS)
- subroutine, private r_ctrper (XDONT, PCLS)
- subroutine, private i_ctrper (XDONT, PCLS)
- real(kind=dp) function, private d_fndnth (XDONT, NORD)
- real(kind=sp) function, private r_fndnth (XDONT, NORD)
- integer(kind=i4) function, private i_fndnth (XDONT, NORD)
- subroutine, private d_indmed (XDONT, INDM)
- recursive subroutine, private d_med (XDATT, IDATT, ires_med)
- subroutine, private r_indmed (XDONT, INDM)
- recursive subroutine, private r_med (XDATT, IDATT, ires_med)
- subroutine, private i_indmed (XDONT, INDM)
- recursive subroutine, private i_med (XDATT, IDATT, ires_med)
- integer(kind=i4) function, private d_indnth (XDONT, NORD)
- integer(kind=i4) function, private r_indnth (XDONT, NORD)
- integer(kind=i4) function, private i_indnth (XDONT, NORD)
- subroutine, private d_inspar (XDONT, NORD)
- subroutine, private r_inspar (XDONT, NORD)
- subroutine, private i_inspar (XDONT, NORD)
- subroutine, private d_inssor (XDONT)
- subroutine, private r_inssor (XDONT)
- subroutine, private i_inssor (XDONT)
- real(kind=dp) function, private d_median (XDONT)
- real(kind=sp) function, private r_median (XDONT)
- integer(kind=i4) function, private i_median (XDONT)
- subroutine, private d_mrgref (XVALT, IRNGT)
- subroutine, private r_mrgref (XVALT, IRNGT)
- subroutine, private i_mrgref (XVALT, IRNGT)
- subroutine, private d_mrgrnk (XDONT, IRNGT)
- subroutine, private r_mrgrnk (XDONT, IRNGT)
- subroutine, private i_mrgrnk (XDONT, IRNGT)
- subroutine, private d_mulcnt (XDONT, IMULT)
- subroutine, private r_mulcnt (XDONT, IMULT)
- subroutine, private i_mulcnt (XDONT, IMULT)
- subroutine, private d_rapknr (XDONT, IRNGT, NORD)
- subroutine, private r_rapknr (XDONT, IRNGT, NORD)
- subroutine, private i_rapknr (XDONT, IRNGT, NORD)
- subroutine, private d_refpar (XDONT, IRNGT, NORD)
- subroutine, private r_refpar (XDONT, IRNGT, NORD)
- subroutine, private i_refpar (XDONT, IRNGT, NORD)
- subroutine, private d_refsor (XDONT)
- recursive subroutine, private d_subsor (XDONT, IDEB1, IFIN1)
- subroutine, private r_refsor (XDONT)
- recursive subroutine, private r_subsor (XDONT, IDEB1, IFIN1)
- subroutine, private i_refsor (XDONT)
- recursive subroutine, private i_subsor (XDONT, IDEB1, IFIN1)
- subroutine, private d_rinpar (XDONT, IRNGT, NORD)
- subroutine, private r_rinpar (XDONT, IRNGT, NORD)
- subroutine, private i_rinpar (XDONT, IRNGT, NORD)
- subroutine, private d_rnkpar (XDONT, IRNGT, NORD)
- subroutine, private r_rnkpar (XDONT, IRNGT, NORD)
- subroutine, private i_rnkpar (XDONT, IRNGT, NORD)
- subroutine, private d_uniinv (XDONT, IGOEST)
- subroutine, private r_uniinv (XDONT, IGOEST)

- subroutine, private i_uniinv (XDONT, IGOEST)
- real(kind=dp) function, private d_nearless (XVAL)
- real(kind=sp) function, private r_nearless (XVAL)
- integer(kind=i4) function, private i_nearless (XVAL)
- subroutine, private d_unipar (XDONT, IRNGT, NORD)
- subroutine, private r_unipar (XDONT, IRNGT, NORD)
- subroutine, private i_unipar (XDONT, IRNGT, NORD)
- subroutine, private d_unirnk (XVALT, IRNGT, NUNI)
- subroutine, private r_unirnk (XVALT, IRNGT, NUNI)
- subroutine, private i_unirnk (XVALT, IRNGT, NUNI)
- subroutine, private d_unista (XDONT, NUNI)
- subroutine, private r_unista (XDONT, NUNI)
- subroutine, private i_unista (XDONT, NUNI)
- recursive real(kind=dp)
  function, private d_valmed (XDONT)
- recursive real(kind=sp)
  function, private r_valmed (XDONT)
- recursive integer(kind=i4)
  function, private i_valmed (XDONT)
- real(kind=dp) function, private d_valnth (XDONT, NORD)
- real(kind=sp) function, private r_valnth (XDONT, NORD)
- integer(kind=i4) function, private i_valnth (XDONT, NORD)

Private Attributes

- integer(kind=i4), dimension(:),
  allocatable, save idont

15.118.1 Detailed Description

Sort and ranking routines.

This module is the Orderpack 2.0 from Michel Olagnon. It provides order and unconditional, unique, and partial ranking, sorting, and permutation.

Authors

Michel Olagnon

Date

2000-2012

Definition at line 12 of file mo_orderpack.f90.

15.118.2 Member Function/Subroutine Documentation

15.118.2.1 subroutine, private mo_orderpack::d_ctrper (real(kind = dp), dimension (:), intent(inout) *XDONT,* real(kind = dp), intent(in) *PCLS*) [private]

Definition at line 427 of file mo_orderpack.f90.

15.118.2.2 real(kind = dp) function, private mo_orderpack::d_fndnth (real(kind = dp), dimension (:), intent(in) *XDONT,* integer(kind = i4), intent(in) *NORD*) [private]

Definition at line 529 of file mo_orderpack.f90.

15.118.2.3 subroutine, private mo_orderpack::d_indmed (real(kind = dp), dimension (:), intent(in) *XDONT,* integer(kind = i4), intent(out) *INDM*) [private]

Definition at line 682 of file mo_orderpack.f90.

References d_med().

Here is the call graph for this function:

15.118.2.4 integer(kind = i4) function, private mo_orderpack::d_indnth (real(kind = dp), dimension (:), intent(in) *XDONT,* integer(kind = i4), intent(in) *NORD*) [private]

Definition at line 1876 of file mo_orderpack.f90.

15.118.2.5 subroutine, private mo_orderpack::d_inspar (real(kind = dp), dimension (:), intent(inout) *XDONT,* integer(kind = i4), intent(in) *NORD*) [private]

Definition at line 3394 of file mo_orderpack.f90.

15.118.2.6 subroutine, private mo_orderpack::d_inssor (real(kind = dp), dimension (:), intent(inout) *XDONT*) [private]

Definition at line 3529 of file mo_orderpack.f90.

15.118.2.7 recursive subroutine, private mo_orderpack::d_med (real(kind = dp), dimension (:), intent(in) *XDATT,* integer(kind = i4), dimension (:), intent(in) *IDATT,* integer(kind = i4), intent(out) *ires_med*) [private]

Definition at line 700 of file mo_orderpack.f90.

Referenced by d_indmed().

Here is the caller graph for this function:

15.118.2.8 real(kind = dp) function, private mo_orderpack::d_median (real(kind = dp), dimension (:), intent(in) *XDONT*) [private]

Definition at line 3721 of file mo_orderpack.f90.

15.118.2.9 subroutine, private mo_orderpack::d_mrgref (real(kind = dp), dimension (:), intent(in) *XVALT,* integer(kind = i4), dimension (:), intent(out) *IRNGT*) [private]

Definition at line 5228 of file mo_orderpack.f90.

15.118.2.10 subroutine, private mo_orderpack::d_mrgrnk (real(kind = dp), dimension (:), intent(in) *XDONT,* integer(kind = i4), dimension (:), intent(out) *IRNGT*) [private]

Definition at line 5615 of file mo_orderpack.f90.

15.118.2.11 subroutine, private mo_orderpack::d_mulcnt (real(kind = dp), dimension (:), intent(in) *XDONT,* integer(kind = i4), dimension (:), intent(out) *IMULT*) [private]

Definition at line 6215 of file mo_orderpack.f90.

15.118.2.12 real(kind = dp) function, private mo_orderpack::d_nearless (real(kind = dp), intent(in) *XVAL*) [private]

Definition at line 10971 of file mo_orderpack.f90.

15.118.2.13 subroutine, private mo_orderpack::d_rapknr (real(kind = dp), dimension (:), intent(in) *XDONT,* integer(kind = i4), dimension (:), intent(out) *IRNGT,* integer(kind = i4), intent(in) *NORD*) [private]

Definition at line 6296 of file mo_orderpack.f90.

15.118.2.14 subroutine, private mo_orderpack::d_refpar (real(kind = dp), dimension (:), intent(in) *XDONT,* integer(kind = i4), dimension (:), intent(out) *IRNGT,* integer(kind = i4), intent(in) *NORD*) [private]

Definition at line 7861 of file mo_orderpack.f90.

15.118.2.15 subroutine, private mo_orderpack::d_refsor (real(kind = dp), dimension (:), intent(inout) *XDONT*) [private]

Definition at line 8211 of file mo_orderpack.f90.

References mo_orderpack::inssor::d_inssor(), and d_subsor().

Here is the call graph for this function:

15.118.2.16 subroutine, private mo_orderpack::d_rinpar (real(kind = dp), dimension (:), intent(in) *XDONT,* integer(kind = i4), dimension (:), intent(out) *IRNGT,* integer(kind = i4), intent(in) *NORD*) `[private]`

Definition at line 8535 of file mo_orderpack.f90.

15.118.2.17 subroutine, private mo_orderpack::d_rnkpar (real(kind = dp), dimension (:), intent(in) *XDONT,* integer(kind = i4), dimension (:), intent(out) *IRNGT,* integer(kind = i4), intent(in) *NORD*) `[private]`

Definition at line 8670 of file mo_orderpack.f90.

15.118.2.18 recursive subroutine, private mo_orderpack::d_subsor (real(kind = dp), dimension (:), intent(inout) *XDONT,* integer(kind = i4), intent(in) *IDEB1,* integer(kind = i4), intent(in) *IFIN1*) `[private]`

Definition at line 8235 of file mo_orderpack.f90.

Referenced by d_refsor().

Here is the caller graph for this function:

15.118.2.19 subroutine, private mo_orderpack::d_uniinv (real(kind = dp), dimension (:), intent(in) *XDONT,* integer(kind = i4), dimension (:), intent(out) *IGOEST*) `[private]`

Definition at line 10230 of file mo_orderpack.f90.

15.118.2.20 subroutine, private mo_orderpack::d_unipar (real(kind = dp), dimension (:), intent(in) *XDONT,* integer(kind = i4), dimension (:), intent(out) *IRNGT,* integer(kind = i4), intent(inout) *NORD*) `[private]`

Definition at line 11005 of file mo_orderpack.f90.

15.118.2.21 subroutine, private mo_orderpack::d_unirnk (real(kind = dp), dimension (:), intent(in) *XVALT,* integer(kind = i4), dimension (:), intent(out) *IRNGT,* integer(kind = i4), intent(out) *NUNI*) `[private]`

Definition at line 12808 of file mo_orderpack.f90.

15.118.2.22 subroutine, private mo_orderpack::d_unista (real(kind = dp), dimension (:), intent(inout) *XDONT,* integer(kind = i4), intent(out) *NUNI*) `[private]`

Definition at line 13553 of file mo_orderpack.f90.

15.118.2.23 recursive real(kind = dp) function, private mo_orderpack::d_valmed (real(kind = dp), dimension (:), intent(in) *XDONT*) `[private]`

Definition at line 13640 of file mo_orderpack.f90.

References mo_orderpack::valmed::d_valmed().

Here is the call graph for this function:

15.118.2.24 real(kind = dp) function, private mo_orderpack::d_valnth (real(kind = dp), dimension (:), intent(in) *XDONT,* integer(kind = i4), intent(in) *NORD*) `[private]`

Definition at line 14648 of file mo_orderpack.f90.

15.118.2.25 subroutine, private mo_orderpack::i_ctrper (integer(kind = i4), dimension (:), intent(inout) *XDONT,* real(kind = sp), intent(in) *PCLS*) `[private]`

Definition at line 495 of file mo_orderpack.f90.

15.118.2.26 integer(kind = i4) function, private mo_orderpack::i_fndnth (integer(kind = i4), dimension (:), intent(in) *XDONT,* integer(kind = i4), intent(in) *NORD*) `[private]`

Definition at line 631 of file mo_orderpack.f90.

15.118.2.27 subroutine, private mo_orderpack::i_indmed (integer(kind = i4), dimension (:), intent(in) *XDONT,* integer(kind = i4), intent(out) *INDM*) `[private]`

Definition at line 1478 of file mo_orderpack.f90.

References i_med().

Here is the call graph for this function:

15.118.2.28 integer(kind = i4) function, private mo_orderpack::i_indnth (integer(kind = i4), dimension (:), intent(in) *XDONT,* integer(kind = i4), intent(in) *NORD*) `[private]`

Definition at line 2888 of file mo_orderpack.f90.

15.118.2.29 subroutine, private mo_orderpack::i_inspar (integer(kind = i4), dimension (:), intent(inout) *XDONT,* integer(kind = i4), intent(in) *NORD*) `[private]`

Definition at line 3484 of file mo_orderpack.f90.

15.118.2.30 subroutine, private mo_orderpack::i_inssor (integer(kind = i4), dimension (:), intent(inout) *XDONT*) `[private]`

Definition at line 3657 of file mo_orderpack.f90.

15.118.2.31 recursive subroutine, private mo_orderpack::i_med (integer(kind = i4), dimension (:), intent(in) *XDATT,* integer(kind = i4), dimension (:), intent(in) *IDATT,* integer(kind = i4), intent(out) *ires_med*) `[private]`

Definition at line 1496 of file mo_orderpack.f90.

Referenced by i_indmed().

Here is the caller graph for this function:

15.118.2.32 integer(kind = i4) function, private mo_orderpack::i_median (integer(kind = i4), dimension (:), intent(in) *XDONT*) [private]

Definition at line 4726 of file mo_orderpack.f90.

15.118.2.33 subroutine, private mo_orderpack::i_mrgref (integer(kind = i4), dimension (:), intent(in) *XVALT,* integer(kind = i4), dimension (:), intent(out) *IRNGT*) [private]

Definition at line 5486 of file mo_orderpack.f90.

15.118.2.34 subroutine, private mo_orderpack::i_mrgrnk (integer(kind = i4), dimension (:), intent(in) *XDONT,* integer(kind = i4), dimension (:), intent(out) *IRNGT*) [private]

Definition at line 6015 of file mo_orderpack.f90.

15.118.2.35 subroutine, private mo_orderpack::i_mulcnt (integer(kind = i4), dimension (:), intent(in) *XDONT,* integer(kind = i4), dimension (:), intent(out) *IMULT*) [private]

Definition at line 6269 of file mo_orderpack.f90.

15.118.2.36 integer(kind = i4) function, private mo_orderpack::i_nearless (integer(kind = i4), intent(in) *XVAL*) [private]

Definition at line 10993 of file mo_orderpack.f90.

15.118.2.37 subroutine, private mo_orderpack::i_rapknr (integer(kind = i4), dimension (:), intent(in) *XDONT,* integer(kind = i4), dimension (:), intent(out) *IRNGT,* integer(kind = i4), intent(in) *NORD*) [private]

Definition at line 7339 of file mo_orderpack.f90.

15.118.2.38 subroutine, private mo_orderpack::i_refpar (integer(kind = i4), dimension (:), intent(in) *XDONT,* integer(kind = i4), dimension (:), intent(out) *IRNGT,* integer(kind = i4), intent(in) *NORD*) [private]

Definition at line 8095 of file mo_orderpack.f90.

15.118.2.39 subroutine, private mo_orderpack::i_refsor (integer(kind = i4), dimension (:), intent(inout) *XDONT*) [private]

Definition at line 8427 of file mo_orderpack.f90.

References mo_orderpack::inssor::i_inssor(), and i_subsor().

Here is the call graph for this function:

15.118.2.40 subroutine, private mo_orderpack::i_rinpar (integer(kind = i4), dimension (:), intent(in) *XDONT,* integer(kind = i4), dimension (:), intent(out) *IRNGT,* integer(kind = i4), intent(in) *NORD*) [private]

Definition at line 8625 of file mo_orderpack.f90.

15.118.2.41 subroutine, private mo_orderpack::i_rnkpar (integer(kind = i4), dimension (:), intent(in) *XDONT,* integer(kind = i4), dimension (:), intent(out) *IRNGT,* integer(kind = i4), intent(in) *NORD*) [private]

Definition at line 9710 of file mo_orderpack.f90.

15.118.2.42 recursive subroutine, private mo_orderpack::i_subsor (integer(kind = i4), dimension (:), intent(inout) *XDONT,* integer(kind = i4), intent(in) *IDEB1,* integer(kind = i4), intent(in) *IFIN1*) [private]

Definition at line 8451 of file mo_orderpack.f90.

Referenced by i_refsor().

Here is the caller graph for this function:

15.118.2.43 subroutine, private mo_orderpack::i_uniinv (integer(kind = i4), dimension (:), intent(in) *XDONT,* integer(kind = i4), dimension (:), intent(out) *IGOEST*) [private]

Definition at line 10724 of file mo_orderpack.f90.

15.118.2.44 subroutine, private mo_orderpack::i_unipar (integer(kind = i4), dimension (:), intent(in) *XDONT,* integer(kind = i4), dimension (:), intent(out) *IRNGT,* integer(kind = i4), intent(inout) *NORD*) [private]

Definition at line 12207 of file mo_orderpack.f90.

15.118.2.45 subroutine, private mo_orderpack::i_unirnk (integer(kind = i4), dimension (:), intent(in) *XVALT,* integer(kind = i4), dimension (:), intent(out) *IRNGT,* integer(kind = i4), intent(out) *NUNI*) [private]

Definition at line 13304 of file mo_orderpack.f90.

15.118.2.46 subroutine, private mo_orderpack::i_unista (integer(kind = i4), dimension (:), intent(inout) *XDONT,* integer(kind = i4), intent(out) *NUNI*) [private]

Definition at line 13611 of file mo_orderpack.f90.

15.118.2.47 recursive integer(kind = i4) function, private mo_orderpack::i_valmed (integer(kind = i4), dimension (:), intent(in) *XDONT*) [private]

Definition at line 14312 of file mo_orderpack.f90.

References mo_orderpack::valmed::i_valmed().

Here is the call graph for this function:

15.118.2.48 integer(kind = i4) function, private mo_orderpack::i_valnth (integer(kind = i4), dimension (:), intent(in) *XDONT,* integer(kind = i4), intent(in) *NORD*) [private]

Definition at line 15634 of file mo_orderpack.f90.

15.118.2.49 subroutine, private mo_orderpack::r_ctrper (real(kind = sp), dimension (:), intent(inout) *XDONT,* real(kind = sp), intent(in) *PCLS*) [private]

Definition at line 461 of file mo_orderpack.f90.

15.118.2.50 real(kind = sp) function, private mo_orderpack::r_fndnth (real(kind = sp), dimension (:), intent(in) *XDONT,* integer(kind = i4), intent(in) *NORD*) [private]

Definition at line 580 of file mo_orderpack.f90.

15.118.2.51 subroutine, private mo_orderpack::r_indmed (real(kind = sp), dimension (:), intent(in) *XDONT,* integer(kind = i4), intent(out) *INDM*) [private]

Definition at line 1080 of file mo_orderpack.f90.

References r_med().

Here is the call graph for this function:

15.118.2.52 integer(kind = i4) function, private mo_orderpack::r_indnth (real(kind = sp), dimension (:), intent(in) *XDONT,* integer(kind = i4), intent(in) *NORD*) [private]

Definition at line 2382 of file mo_orderpack.f90.

15.118.2.53 subroutine, private mo_orderpack::r_inspar (real(kind = sp), dimension (:), intent(inout) *XDONT,* integer(kind = i4), intent(in) *NORD*) [private]

Definition at line 3439 of file mo_orderpack.f90.

15.118.2.54 subroutine, private mo_orderpack::r_inssor (real(kind = sp), dimension (:), intent(inout) *XDONT*) [private]

Definition at line 3593 of file mo_orderpack.f90.

15.118.2.55 recursive subroutine, private mo_orderpack::r_med (real(kind = sp), dimension (:), intent(in) *XDATT,* integer(kind = i4), dimension (:), intent(in) *IDATT,* integer(kind = i4), intent(out) *ires_med*) [private]

Definition at line 1098 of file mo_orderpack.f90.

Referenced by r_indmed().

Here is the caller graph for this function:

15.118.2.56 real(kind = sp) function, private mo_orderpack::r_median (real(kind = sp), dimension (:), intent(in) *XDONT*) [private]

Definition at line 4224 of file mo_orderpack.f90.

15.118.2.57 subroutine, private mo_orderpack::r_mrgref (real(kind = sp), dimension (:), intent(in) *XVALT,* integer(kind = i4), dimension (:), intent(out) *IRNGT*) [private]

Definition at line 5357 of file mo_orderpack.f90.

15.118.2.58 subroutine, private mo_orderpack::r_mrgrnk (real(kind = sp), dimension (:), intent(in) *XDONT,* integer(kind = i4), dimension (:), intent(out) *IRNGT*) [private]

Definition at line 5815 of file mo_orderpack.f90.

15.118.2.59 subroutine, private mo_orderpack::r_mulcnt (real(kind = sp), dimension (:), intent(in) *XDONT,* integer(kind = i4), dimension (:), intent(out) *IMULT*) [private]

Definition at line 6242 of file mo_orderpack.f90.

15.118.2.60 real(kind = sp) function, private mo_orderpack::r_nearless (real(kind = sp), intent(in) *XVAL*) [private]

Definition at line 10982 of file mo_orderpack.f90.

15.118.2.61 subroutine, private mo_orderpack::r_rapknr (real(kind = sp), dimension (:), intent(in) *XDONT,* integer(kind = i4), dimension (:), intent(out) *IRNGT,* integer(kind = i4), intent(in) *NORD*) [private]

Definition at line 6817 of file mo_orderpack.f90.

15.118.2.62 subroutine, private mo_orderpack::r_refpar (real(kind = sp), dimension (:), intent(in) *XDONT,* integer(kind = i4), dimension (:), intent(out) *IRNGT,* integer(kind = i4), intent(in) *NORD*) [private]

Definition at line 7978 of file mo_orderpack.f90.

15.118.2.63 subroutine, private mo_orderpack::r_refsor (real(kind = sp), dimension (:), intent(inout) *XDONT*) [private]

Definition at line 8319 of file mo_orderpack.f90.

References mo_orderpack::inssor::r_inssor(), and r_subsor().

Here is the call graph for this function:

15.118.2.64 subroutine, private mo_orderpack::r_rinpar (real(kind = sp), dimension (:), intent(in) *XDONT,* integer(kind = i4), dimension (:), intent(out) *IRNGT,* integer(kind = i4), intent(in) *NORD*) [private]

Definition at line 8580 of file mo_orderpack.f90.

15.118.2.65 subroutine, private mo_orderpack::r_rnkpar (real(kind = sp), dimension (:), intent(in) *XDONT,* integer(kind = i4), dimension (:), intent(out) *IRNGT,* integer(kind = i4), intent(in) *NORD*) [private]

Definition at line 9190 of file mo_orderpack.f90.

15.118.2.66 recursive subroutine, private mo_orderpack::r_subsor (real(kind = sp), dimension (:), intent(inout) *XDONT,* integer(kind = i4), intent(in) *IDEB1,* integer(kind = i4), intent(in) *IFIN1*) [private]

Definition at line 8343 of file mo_orderpack.f90.

Referenced by r_refsor().

Here is the caller graph for this function:

15.118.2.67 subroutine, private mo_orderpack::r_uniinv (real(kind = sp), dimension (:), intent(in) *XDONT,* integer(kind = i4), dimension (:), intent(out) *IGOEST*) [private]

Definition at line 10477 of file mo_orderpack.f90.

15.118.2.68 subroutine, private mo_orderpack::r_unipar (real(kind = sp), dimension (:), intent(in) *XDONT,* integer(kind = i4), dimension (:), intent(out) *IRNGT,* integer(kind = i4), intent(inout) *NORD*) [private]

Definition at line 11606 of file mo_orderpack.f90.

15.118.2.69 subroutine, private mo_orderpack::r_unirnk (real(kind = sp), dimension (:), intent(in) *XVALT,* integer(kind = i4), dimension (:), intent(out) *IRNGT,* integer(kind = i4), intent(out) *NUNI*) [private]

Definition at line 13056 of file mo_orderpack.f90.

15.118.2.70 subroutine, private mo_orderpack::r_unista (real(kind = sp), dimension (:), intent(inout) *XDONT,* integer(kind = i4), intent(out) *NUNI*) [private]

Definition at line 13582 of file mo_orderpack.f90.

15.118.2.71 recursive real(kind = sp) function, private mo_orderpack::r_valmed (real(kind = sp), dimension (:), intent(in) *XDONT*) [private]

Definition at line 13976 of file mo_orderpack.f90.

References mo_orderpack::valmed::r_valmed().

Here is the call graph for this function:

15.118.2.72 real(kind = sp) function, private mo_orderpack::r_valnth (real(kind = sp), dimension (:), intent(in) *XDONT,* integer(kind = i4), intent(in) *NORD*) [private]

Definition at line 15141 of file mo_orderpack.f90.

15.118.2.73 integer(i4) function, dimension(size(arr)) mo_orderpack::sort_index_dp (real(dp), dimension(:), intent(in) *arr*) [private]

Definition at line 391 of file mo_orderpack.f90.

References mo_orderpack::sort_index::sort_index_dp().

Here is the call graph for this function:

15.118.2.74 integer(i4) function, dimension(size(arr)) mo_orderpack::sort_index_i4 (integer(i4), dimension(:), intent(in) *arr*) [private]

Definition at line 413 of file mo_orderpack.f90.

References mo_orderpack::sort_index::sort_index_i4().

Here is the call graph for this function:

15.118.2.75 integer(i4) function, dimension(size(arr)) mo_orderpack::sort_index_sp (real(sp), dimension(:), intent(in) *arr*) [private]

Definition at line 402 of file mo_orderpack.f90.

References mo_orderpack::sort_index::sort_index_sp().

Here is the call graph for this function:

15.118.3 Member Data Documentation

15.118.3.1 integer(kind = i4), dimension(:), allocatable, save mo_orderpack::idont [private]

Definition at line 385 of file mo_orderpack.f90.

The documentation for this module was generated from the following file:

- mo_orderpack.f90

15.119 mo_percentile Module Reference

Data Types

- interface median
- interface n_element
- interface percentile
- interface qmedian

Private Member Functions

- real(dp) function median_dp (arrin, mask)
- real(sp) function median_sp (arrin, mask)
- real(dp) function n_element_dp (idat, n, mask, before, after, previous, next)
- real(sp) function n_element_sp (idat, n, mask, before, after, previous, next)
- real(dp) function percentile_0d_dp (arrin, k, mask, mode_in)
- real(sp) function percentile_0d_sp (arrin, k, mask, mode_in)
- real(dp) function, dimension(size(k)) percentile_1d_dp (arrin, k, mask, mode_in)
- real(sp) function, dimension(size(k)) percentile_1d_sp (arrin, k, mask, mode_in)
- real(dp) function qmedian_dp (dat)
- real(sp) function qmedian_sp (dat)

15.119.1 Detailed Description

Definition at line 1 of file mo_percentile.f90.

15.119.2 Member Function/Subroutine Documentation

15.119.2.1 real(dp) function mo_percentile::median_dp (real(dp), dimension(:), intent(in) *arrin*, logical, dimension(:), intent(in), optional *mask*) [private]

Definition at line 281 of file mo_percentile.f90.

References mo_percentile::median::median_dp().

Here is the call graph for this function:

15.119.2.2 real(sp) function mo_percentile::median_sp (real(sp), dimension(:), intent(in) *arrin,* logical, dimension(:), intent(in), optional *mask*) [private]

Definition at line 323 of file mo_percentile.f90.

References mo_percentile::median::median_sp().

Here is the call graph for this function:

15.119.2.3 real(dp) function mo_percentile::n_element_dp (real(dp), dimension(:), intent(in) *idat,* integer(i4), intent(in) *n,* logical, dimension(:), intent(in), optional *mask,* real(dp), intent(out), optional *before,* real(dp), intent(out), optional *after,* real(dp), intent(out), optional *previous,* real(dp), intent(out), optional *next*) [private]

Definition at line 366 of file mo_percentile.f90.

References mo_percentile::n_element::n_element_dp().

Here is the call graph for this function:

15.119.2.4 real(sp) function mo_percentile::n_element_sp (real(sp), dimension(:), intent(in) *idat,* integer(i4), intent(in) *n,* logical, dimension(:), intent(in), optional *mask,* real(sp), intent(out), optional *before,* real(sp), intent(out), optional *after,* real(sp), intent(out), optional *previous,* real(sp), intent(out), optional *next*) [private]

Definition at line 441 of file mo_percentile.f90.

References mo_percentile::n_element::n_element_sp().

Here is the call graph for this function:

15.119.2.5 real(dp) function mo_percentile::percentile_0d_dp (real(dp), dimension(:), intent(in) *arrin,* real(dp), intent(in) *k,* logical, dimension(:), intent(in), optional *mask,* integer(i4), intent(in), optional *mode_in*) [private]

Definition at line 518 of file mo_percentile.f90.

References mo_percentile::percentile::percentile_0d_dp().

Here is the call graph for this function:

15.119.2.6 real(sp) function mo_percentile::percentile_0d_sp (real(sp), dimension(:), intent(in) *arrin,* real(sp), intent(in) *k,* logical, dimension(:), intent(in), optional *mask,* integer(i4), intent(in), optional *mode_in*) [private]

Definition at line 629 of file mo_percentile.f90.

References mo_percentile::percentile::percentile_0d_sp().

Here is the call graph for this function:

15.119.2.7 real(dp) function, dimension(size(k)) mo_percentile::percentile_1d_dp (real(dp), dimension(:), intent(in) *arrin,* real(dp), dimension(:), intent(in) *k,* logical, dimension(:), intent(in), optional *mask,* integer(i4), intent(in), optional *mode_in*) [private]

Definition at line 740 of file mo_percentile.f90.

References mo_percentile::percentile::percentile_1d_dp().

Here is the call graph for this function:

15.119.2.8 real(sp) function, dimension(size(k)) mo_percentile::percentile_1d_sp (real(sp), dimension(:), intent(in) *arrin,* real(sp), dimension(:), intent(in) *k,* logical, dimension(:), intent(in), optional *mask,* integer(i4), intent(in), optional *mode_in*) [private]

Definition at line 860 of file mo_percentile.f90.

References mo_percentile::percentile::percentile_1d_sp().

Here is the call graph for this function:

15.119.2.9 real(dp) function mo_percentile::qmedian_dp (real(dp), dimension(:), intent(inout) *dat*) [private]

Definition at line 981 of file mo_percentile.f90.

References mo_percentile::qmedian::qmedian_dp().

Here is the call graph for this function:

15.119.2.10 real(sp) function mo_percentile::qmedian_sp (real(sp), dimension(:), intent(inout) *dat*) [private]

Definition at line 1028 of file mo_percentile.f90.

References mo_percentile::qmedian::qmedian_sp().

Here is the call graph for this function:

The documentation for this module was generated from the following file:

- mo_percentile.f90

15.120 mo_pet Module Reference

Module for calculating reference/potential evapotranspiration [mm s-1].

Public Member Functions

- 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.

Private Member Functions

- elemental pure real(dp)
  function, private extraterr_rad_approx (doy, latitude)

  Approximation of extraterrestrial radiation.
- elemental pure real(dp)
  function, private slope_satpressure (tavg)

  slope of saturation vapour pressure curve
- elemental pure real(dp)
  function, private sat_vap_pressure (tavg)

  calculation of the saturation vapour pressure

15.120.1 Detailed Description

Module for calculating reference/potential evapotranspiration [mm s-1].

This module calculates PET [mm/s] based on one of the methods

- Hargreaves-Samani (1982)
- Priestly-Taylor (1972)
- Penman-Monteith FAO (1998)

Authors

Matthias Zink, Christoph Schneider, Matthias Cuntz

Date

Apr 2014

Definition at line 16 of file mo_pet.f90.

15.120.2 Member Function/Subroutine Documentation

15.120.2.1 elemental pure real(dp) function, private mo_pet::extraterr_rad_approx (integer(i4), intent(in) *doy*, real(dp), intent(in) *latitude*) [private]

Approximation of extraterrestrial radiation.

Approximation of extraterrestrial radiation at the top of the atmosphere R_a after Duffie and Beckman (1980). R_a is converted from $[J\,m^{-2}\,d^{-1}]$ in $[mm\,d^{-1}]$.

$$R_a = \frac{86400}{\pi \cdot \lambda} \cdot E_0 \cdot d_r \cdot (\omega \cdot \sin(latitude) \cdot \sin(\delta) + \cos(latitude) \cdot \cos(\delta) \cdot \sin(\omega)$$

where $E_0 = 1367\,J\,m^{-2}\,s^{-1}$ is the solar constant and It is dependent on the following sub equations: The relative distance Earth-Sun:

$$d_r = 1 + 0.033 \cdot \cos\left(\frac{2 \cdot \pi \cdot doy}{365}\right)$$

in which doy is the day of the year. The solar declination [radians] defined by

$$\delta = 0.4093 \cdot \sin\left(\frac{2 \cdot \pi \cdot doy}{365} - 1.405\right)$$

The sunset hour angle [radians]:

$$\omega = \arccos(-\tan(latitude) * \tan(\delta))$$

$< \lambda = 2.45 \cdot 10^6\,J\,m^{-2}\,mm^{-1}$ is the latent heat of vaporization.

Parameters

in	*integer(i4) :: doy*	day of year [-]
in	*real(dp) :: latitude*	latitude [rad]

Returns

real(dp) :: extraterr_rad_approx — extraterrestrial radiation approximation $[W\ m^{-2}]$

Authors

Matthias Zink

Date

Apr 2014

Definition at line 318 of file mo_pet.f90.

Referenced by pet_hargreaves().

Here is the caller graph for this function:

15.120.2.2 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 $[°C]$
in	*real(dp) :: tmax*	daily maximum of temp $[°C]$
in	*real(dp) :: tmin*	daily minimum of temp $[°C]$
in	*real(dp) :: latitude*	latitude of the cell for Ra estimation $[radians]$
in	*integer(i4) :: doy*	day of year for Ra estimation

Returns

real(dp) :: pet_hargreaves — Hargreaves-Samani pot. evapotranspiration [mm s-1]

Authors

Matthias Zink

Date

Dec 2012

Definition at line 75 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:

15.120.2.3 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 $[^{\circ}C]$
in	*real(dp) :: act_↩ vap_pressure*	actual vapur pressure $[kPa]$
in	*real(dp) :: aerodyn_↩ resistance*	aerodynmaical resistance $s\,m^{-1}$
in	*real(dp) :: bulksurface_↩ resistance*	bulk surface resistance $s\,m^{-1}$
in	*real(dp) :: a_s*	fraction of one-sided leaf area covered by stomata 1
in	*real(dp) :: a_sh*	fraction of projected area exchanging sensible heat with the air 1

Returns

real(dp) :: pet_penman — Reference Evapotranspiration [mm s-1]

Authors

Matthias Zink

Date

Apr 2014

Definition at line 238 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:

15.120.2.4 elemental pure real(dp) function, public mo_pet::pet_priestly (real(dp), intent(in) *PrieTayParam,* real(dp), intent(in) *Rn,* real(dp), intent(in) *tavg*)

Reference Evapotranspiration after Priestly-Taylor.

Calculates the Reference Evapotranspiration $[mm\ d^{-1}]$ based on the Priestly-Taylor (1972) model for every given cell by applying the equation

$$PET = \alpha * \frac{\Delta}{(\gamma + \Delta)} * R_n$$

where R_n $[W\ m^{-2}]$ is the net solar radiation $\Delta = f(T_{avg})$ is the slope of the saturation-vapour pressure curve and α is a emperical coefficient.

Parameters

in	*real(dp) :: PrieTayParam*	Priestley-Taylor coefficient $\alpha[-]$
in	*real(dp) :: Rn*	net solar radiation $[W\ m^{-2}]$
in	*real(dp) :: Tavg*	

Returns

real(dp) :: pet_priestly — Priestley-Taylor pot. evapotranspiration [mm s-1]

Authors

Matthias Zink

Date

Apr 2014

Definition at line 151 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:

15.120.2.5 elemental pure real(dp) function, private mo_pet::sat_vap_pressure (real(dp), intent(in) *tavg*) [private]

calculation of the saturation vapour pressure

Calculation of the saturation vapour pressure

$$e_s(T_a) = 0.6108 \cdot \exp\left(\frac{17.27 \cdot T_a}{T_a + 237.3}\right)$$

Parameters

in	*real(dp) :: tavg*	temperature [degC]

Returns

real(dp) :: sat_vap_pressure — saturation vapour pressure [kPa]

Authors

Matthias Zink

Date

Apr 2014

Definition at line 431 of file mo_pet.f90.

Referenced by pet_penman(), and slope_satpressure().

Here is the caller graph for this function:

15.120.2.6 elemental pure real(dp) function, private mo_pet::slope_satpressure (real(dp), intent(in) *tavg*) [private]

slope of saturation vapour pressure curve

slope of saturation vapour pressure curve after Tetens

$$\Delta = \frac{0.6108 * e_s(T_a)}{e^{(}2 \cdot \log(T_a + 237.3))}$$

Parameters

in	*real(dp) :: tavg*	average daily temperature $[^\circ C]$

Returns

real(dp) :: slope_satpressure — slope of saturation vapour pressure curve $[kPa\ K-1]$

Authors

Matthias Zink

Date

Apr 2014

Definition at line 389 of file mo_pet.f90.

References sat_vap_pressure().

Referenced by pet_penman(), and pet_priestly().

Here is the call graph for this function:

Here is the caller graph for this function:

The documentation for this module was generated from the following file:

- mo_pet.f90

15.121 mo_read_forcing_nc Module Reference

Reads forcing input data.

Public Member Functions

- subroutine, public read_forcing_nc (folder, nRows, nCols, varName, mask, data, target_period, lower, upper, nctimestep, fileName, nocheck, maskout)

  Reads forcing input in NetCDF file format.
- subroutine, public read_const_forcing_nc (folder, nRows, nCols, varName, mask, data)
- subroutine, public read_weights_nc (folder, nRows, nCols, varName, data, mask, lower, upper, nocheck, maskout, fileName)

  Reads weights for meteo forcings input in NetCDF file format.
- subroutine get_time_vector_and_select (var, fname, inctimestep, time_start, time_cnt, target_period)

  TODO: add description.

15.121.1 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

Definition at line 22 of file mo_read_forcing_nc.f90.

15.121.2 Member Function/Subroutine Documentation

15.121.2.1 subroutine mo_read_forcing_nc::get_time_vector_and_select (type(ncvariable), intent(in) *var,* character(256), intent(in) *fname,* integer(i4), intent(in) *inctimestep,* integer(i4), intent(out) *time_start,* integer(i4), intent(out) *time_cnt,* type(period), intent(in), optional *target_period*)

TODO: add description.

TODO: add description ADDITIONAL INFORMATION get_time_vector_and_select Extract time vector in unit julian hours and get supposed time step in hours

Parameters

in	*type(NcVariable) :: var*	variable of interest
in	*character(256) :: fname*	fname of ncfile for error message
in	*integer(i4) :: inctimestep*	flag for requested time step
out	*integer(i4) :: time_start*	time_start index of time selection
out	*integer(i4) :: time_cnt*	time_count of indexes of time selection
in	*type(period), optional :: target_period*	reference period

Authors

Matthias Zink

Date

Oct 2012

Definition at line 595 of file mo_read_forcing_nc.f90.

References mo_julian::caldat(), mo_julian::dec2date(), mo_string_utils::divide_string(), mo_julian::julday(), and mo_message::message().

Referenced by read_forcing_nc().

Here is the call graph for this function:

Here is the caller graph for this function:

15.121.2.2 subroutine, public mo_read_forcing_nc::read_const_forcing_nc (character(len=∗), intent(in) *folder,* integer(i4), intent(in) *nRows,* integer(i4), intent(in) *nCols,* character(len=∗), intent(in) *varName,* logical, dimension(:,:), intent(in) *mask,* real(dp), dimension(:,:), intent(out), allocatable *data*)

Parameters

in	*folder*	

Author

Lennart Schueler, heavily influenced by read_forcing_nc

Date

May 2018

Definition at line 260 of file mo_read_forcing_nc.f90.

References mo_message::message().

Referenced by mo_mrm_read_data::mrm_read_bankfull_runoff().

Here is the call graph for this function:

Here is the caller graph for this function:

15.121.2.3 subroutine, public mo_read_forcing_nc::read_forcing_nc (character(len = ∗), intent(in) *folder,* integer(i4), intent(in) *nRows,* integer(i4), intent(in) *nCols,* character(len = ∗), intent(in) *varName,* logical, dimension(:, :), intent(in) *mask,* real(dp), dimension(:, :, :), intent(out), allocatable *data,* type(period), intent(in), optional *target_period,* real(dp), intent(in), optional *lower,* real(dp), intent(in), optional *upper,* integer(i4), intent(in), optional *nctimestep,* character(256), intent(in), optional *fileName,* logical, intent(in), optional *nocheck,* logical, dimension(:, :, :), intent(out), optional, allocatable *maskout*)

Reads forcing input in NetCDF file format.

Reads netCDF forcing files. First, the dimensions given are cross-checked with header.txt information. Second, the data of the specified period are read from the specified directory. If the optional lower and/or upper bound for the data values is given, the read data are checked for validity. The program is stopped if any value lies out of range. If the optinal argument nocheck is true, the data are not checked for coverage with the input mask. Additionally in this case an mask of vild data points can be received from the routine in maskout.

Parameters

in	*character(len = ∗) :: folder*	Name of the folder where data are stored
in	*integer(i4) :: nRows*	Number of datapoints in longitudinal direction
in	*integer(i4) :: nCols*	Number of datapoints in latitudinal direction
in	*character(len = ∗) :: varName*	Name of variable name to read
in	*logical, dimension(:, :) :: mask*	mask of valid data fields
out	*real(dp), dimension(:, :, :) :: data*	Data matrixdim_1 = longitude, dim_2 = latitude, dim_3 = time
in	*type(period), optional :: target_period*	Period the data are needed for
in	*real(dp), optional :: lower*	Lower bound for check of validity of data values
in	*real(dp), optional :: upper*	Upper bound for check of validity of data values
in	*integer(i4), optional :: nctimestep*	timestep in netcdf file
in	*character(256), optional :: fileName*	name of variable, defaults to fileName
in	*logical, optional :: nocheck*	.TRUE. if check for nodata values deactivateddefault = .FALSE. - check is done
out	*logical, dimension(:, :, :), optional :: maskout*	! mask of validdata points

Authors

Matthias Zink

Date

May 2013

Definition at line 85 of file mo_read_forcing_nc.f90.

References get_time_vector_and_select(), and mo_message::message().

Referenced by mo_meteo_forcings::meteo_forcings_wrapper(), mo_mrm_read_data::mrm_read_total_runoff(), mo_read_optional_data::read_evapotranspiration(), mo_read_optional_data::read_neutrons(), and mo_read_optional_data::read_soil_moisture().

Here is the call graph for this function:

Here is the caller graph for this function:

15.121.2.4 subroutine, public mo_read_forcing_nc::read_weights_nc (character(len = ∗), intent(in) *folder,* integer(i4), intent(in) *nRows,* integer(i4), intent(in) *nCols,* character(len = ∗), intent(in) *varName,* real(dp), dimension(:, :, :, :), intent(out), allocatable *data,* logical, dimension(:, :), intent(in) *mask,* real(dp), intent(in), optional *lower,* real(dp), intent(in), optional *upper,* logical, intent(in), optional *nocheck,* logical, dimension(:, :, :, :), intent(out), optional, allocatable *maskout,* character(256), intent(in), optional *fileName*)

Reads weights for meteo forcings input in NetCDF file format.

Reads netCDF weight files. First, the dimensions given are cross-checked with header.txt information. If the optional lower and/or upper bound for the data values is given, the read data are checked for validity. The program is stopped if any value lies out of range. If the optinal argument nocheck is true, the data are not checked for coverage with the input mask. Additionally in this case an mask of vild data points can be received from the routine in maskout.

Parameters

in	*character(len = ∗) :: folder*	Name of the folder where data are stored
in	*integer(i4) :: nRows*	Number of datapoints in longitudinal direction
in	*integer(i4) :: nCols*	Number of datapoints in latitudinal direction
in	*character(len = ∗) :: varName*	Name of variable name to read
in	*logical, dimension(:, :) :: mask*	mask of valid data fields
out	*real(dp), dimension(:, :, :, :) :: data*	Data matrixdim_1 = longitude, dim_2 = latitude, dim_3 = months, dim_4 = hours
in	*real(dp), optional :: lower*	Lower bound for check of validity of data values
in	*real(dp), optional :: upper*	Upper bound for check of validity of data values
in	*logical, optional :: nocheck*	.TRUE. if check for nodata values deactivateddefault = .FALSE. - check is done
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 415 of file mo_read_forcing_nc.f90.

References mo_message::message().

Referenced by mo_meteo_forcings::meteo_weights_wrapper().

Here is the call graph for this function:

Here is the caller graph for this function:

The documentation for this module was generated from the following file:

- mo_read_forcing_nc.f90

15.122 mo_read_latlon Module Reference

reading latitude and longitude coordinates for each basin

Public Member Functions

- subroutine, public read_latlon (ii, lon_var_name, lat_var_name, level_name, level)

  reads latitude and longitude coordinates

15.122.1 Detailed Description

reading latitude and longitude coordinates for each basin

TODO: add description

Authors

Stephan Thober

Date

Nov 2013

Definition at line 13 of file mo_read_latlon.f90.

15.122.2 Member Function/Subroutine Documentation

15.122.2.1 subroutine, public mo_read_latlon::read_latlon (integer(i4), intent(in) *ii,* character(∗), intent(in) *lon_var_name,* character(∗), intent(in) *lat_var_name,* character(∗), intent(in) *level_name,* type(grid), intent(inout) *level*)

reads latitude and longitude coordinates

reads latitude and longitude coordinates from netcdf file for each basin and appends it to the global variables latitude and longitude.

Parameters

in	*integer(i4) :: ii*	basin indexFile name of the basins must be xxx_latlon.nc, wherexxx is the basin id. Variable names in the netcdf filehave to be 'lat' for latitude and 'lon' for longitude.
in	*character(∗) :: lon_var_name*	
in	*character(∗) :: lat_var_name*	
in	*character(∗) :: level_name*	
in,out	*type(Grid) :: level*	

Authors

Stephan Thober

Date

Nov 2013

Definition at line 65 of file mo_read_latlon.f90.

References mo_message::message().

Referenced by mo_mrm_init::mrm_init(), and mo_mrm_read_data::mrm_read_l0_data().

Here is the call graph for this function:

Here is the caller graph for this function:

The documentation for this module was generated from the following file:

- mo_read_latlon.f90

15.123 mo_read_optional_data Module Reference

Read optional data for mHM calibration.

Public Member Functions

- subroutine, public read_soil_moisture (iBasin)

  Read soil moisture data from NetCDF file for calibration.
- subroutine, public read_basin_avg_tws

  Read basin average TWS timeseries from file, the same way runoff is read.
- subroutine, public read_neutrons (iBasin)

  Read neutrons data from NetCDF file for calibration.
- subroutine, public read_evapotranspiration (iBasin)

  Read evapotranspiration data from NetCDF file for calibration.

15.123.1 Detailed Description

Read optional data for mHM calibration.

Data have to be provided in resolution of the hydrology.

Authors

Matthias Zink

Date

Mar 2015

Definition at line 13 of file mo_read_optional_data.f90.

15.123.2 Member Function/Subroutine Documentation

15.123.2.1 subroutine, public mo_read_optional_data::read_basin_avg_tws ( )

Read basin average TWS timeseries from file, the same way runoff is read.

Read basin average TWS timeseries Allocate global basin_avg_TWS variable that contains the simulated values after the simulation.

Authors

Oldrich Rakovec

Date

Oct 2015

Definition at line 148 of file mo_read_optional_data.f90.

References mo_message::message(), and mo_read_timeseries::read_timeseries().

Referenced by mhm_driver().

Here is the call graph for this function:

Here is the caller graph for this function:

15.123.2.2 subroutine, public mo_read_optional_data::read_evapotranspiration (integer(i4), intent(in) *iBasin*)

Read evapotranspiration data from NetCDF file for calibration.

This routine reads oberved evapotranspiration fields which are used for model calibration. The evapotranspiration file is expected to be called "et.nc" with a variable "et" inside. The data are read only for the evaluation period they are intended to be used for calibration. Evapotranspiration data are only read if one of the corresponding objective functions is chosen.

Parameters

in	*integer(i4) :: iBasin*	Basin Id

Authors

Johannes Brenner

Date

Feb 2017

Definition at line 333 of file mo_read_optional_data.f90.

References mo_message::message(), mo_read_forcing_nc::read_forcing_nc(), mo_timer::timer_get(), mo_timer↩ ::timer_start(), and mo_timer::timer_stop().

Referenced by mhm_driver().

Here is the call graph for this function:

Here is the caller graph for this function:

15.123.2.3 subroutine, public mo_read_optional_data::read_neutrons (integer(i4), intent(in) *iBasin*)

Read neutrons data from NetCDF file for calibration.

This routine reads oberved neutron fields which are used for model calibration. The neutrons file is expected to be called "neutrons.nc" with a variable "neutrons" inside. The data are read only for the evaluation period they are intended to be used for calibration. Neutrons data are only read if one of the corresponding objective functions is chosen.

Parameters

in	*integer(i4) :: iBasin*	Basin Id

Authors

Martin Schroen

Date

Jul 2015

Definition at line 229 of file mo_read_optional_data.f90.

References mo_message::message(), mo_read_forcing_nc::read_forcing_nc(), mo_timer::timer_get(), mo_timer↩ ::timer_start(), and mo_timer::timer_stop().

Referenced by mhm_driver().

Here is the call graph for this function:

Here is the caller graph for this function:

15.123.2.4 subroutine, public mo_read_optional_data::read_soil_moisture (integer(i4), intent(in) *iBasin*)

Read soil moisture data from NetCDF file for calibration.

This routine reads oberved soil moisture fields which are used for model calibration. The soil moisture file is expected to be called "sm.nc" with a variable "sm" inside. The data are read only for the evaluation period they are intended to be used for calibration. Soil moisture data are only read if one of the corresponding objective functions is chosen.

Parameters

in	*integer(i4) :: iBasin*	Basin Id

Authors

Matthias Zink

Date

Mar 2015

Definition at line 52 of file mo_read_optional_data.f90.

References mo_message::message(), mo_read_forcing_nc::read_forcing_nc(), mo_timer::timer_get(), mo_timer↩ ::timer_start(), and mo_timer::timer_stop().

Referenced by mhm_driver().

Here is the call graph for this function:

Here is the caller graph for this function:

The documentation for this module was generated from the following file:

- mo_read_optional_data.f90

15.124 mo_read_spatial_data Module Reference

Reads spatial input data.

Data Types

- interface read_spatial_data_ascii

  Reads spatial data files of ASCII format.

Public Member Functions

- 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.

Private Member Functions

- 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.

15.124.1 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

Definition at line 16 of file mo_read_spatial_data.f90.

15.124.2 Member Function/Subroutine Documentation

15.124.2.1 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 369 of file mo_read_spatial_data.f90.

Referenced by mo_startup::l2_variable_init(), mo_common_read_data::read_dem(), read_spatial_data_ascii_dp(), and read_spatial_data_ascii_i4().

Here is the caller graph for this function:

15.124.2.2 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 93 of file mo_read_spatial_data.f90.

References read_header_ascii().

Here is the call graph for this function:

15.124.2.3 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 230 of file mo_read_spatial_data.f90.

References read_header_ascii().

Here is the call graph for this function:

The documentation for this module was generated from the following file:

- mo_read_spatial_data.f90

15.125 mo_read_timeseries Module Reference

Routines to read files containing timeseries data.

Public Member Functions

- subroutine, public read_timeseries (filename, fileunit, periodStart, periodEnd, optimize, opti_function, data, mask, nMeasPerDay)

  Reads time series in ASCII format.

15.125.1 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

Definition at line 14 of file mo_read_timeseries.f90.

15.125.2 Member Function/Subroutine Documentation

15.125.2.1 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 72 of file mo_read_timeseries.f90.

References mo_julian::julday(), and mo_message::message().

Referenced by mo_mrm_read_data::mrm_read_discharge(), and mo_read_optional_data::read_basin_avg_tws().

Here is the call graph for this function:

Here is the caller graph for this function:

The documentation for this module was generated from the following file:

- mo_read_timeseries.f90

15.126 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.

Public Member Functions

- subroutine, public write_restart_files (OutPath)

  write restart files for each basin
- subroutine, public read_restart_states (iBasin, InPath)

  reads fluxes and state variables from file
- subroutine unpack_field_and_write_1d_i4 (nc, var_name, var_dims, fill_value, data, mask, var_long_name)
- subroutine unpack_field_and_write_1d_dp (nc, var_name, var_dims, fill_value, data, mask, var_long_name)
- subroutine unpack_field_and_write_2d_dp (nc, var_name, var_dims, fill_value, data, mask, var_long_name)
- subroutine unpack_field_and_write_3d_dp (nc, var_name, var_dims, fill_value, data, mask, var_long_name)

15.126.1 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

Definition at line 17 of file mo_restart.f90.

15.126.2 Member Function/Subroutine Documentation

15.126.2.1 subroutine, public mo_restart::read_restart_states (integer(i4), intent(in) *iBasin,* character(256), intent(in) *InPath*)

reads fluxes and state variables from file

read fluxes and state variables from given restart directory and initialises all state variables that are initialized in the subroutine initialise, contained in module mo_startup.

Parameters

in	*integer(i4) :: iBasin*	number of basin
in	*character(256) :: InPath*	Input Path including trailing slash

Authors

Stephan Thober

Date

Apr 2013

Definition at line 328 of file mo_restart.f90.

References mo_runoff::l1_total_runoff().

Referenced by mo_mhm_eval::mhm_eval().

Here is the call graph for this function:

Here is the caller graph for this function:

15.126.2.2 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 788 of file mo_restart.f90.

15.126.2.3 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 743 of file mo_restart.f90.

15.126.2.4 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 833 of file mo_restart.f90.

15.126.2.5 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 890 of file mo_restart.f90.

15.126.2.6 subroutine, public mo_restart::write_restart_files (character(256), dimension(:), intent(in) *OutPath*)

write restart files for each basin

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

Parameters

in	*character(256), dimension(:) :: OutPath*	Output Path for each basin

Authors

Stephan Thober

Date

Jun 2014

Definition at line 98 of file mo_restart.f90.

References mo_runoff::l1_total_runoff(), mo_message::message(), 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:

The documentation for this module was generated from the following file:

- mo_restart.f90

15.127 mo_runoff Module Reference

Runoff generation for the unsaturated zone, saturated zone (or groundwater zone), and runoff accumulation.

Public Member Functions

- 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

15.127.1 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

Definition at line 15 of file mo_runoff.f90.

15.127.2 Member Function/Subroutine Documentation

15.127.2.1 subroutine, public mo_runoff::l1_total_runoff (real(dp), intent(in) *fSealed_area_fraction,* real(dp), intent(in) *fast_interflow,* real(dp), intent(in) *slow_interflow,* real(dp), intent(in) *baseflow,* real(dp), intent(in) *direct_runoff,* real(dp), intent(out) *total_runoff*)

total runoff accumulation at level 1

Accumulates runoff.

$$q_T = (q_0 + q_1 + q_2) * (1 - fSealed) + q_D * fSealed$$

, where fSealed is the fraction of sealed area.

Parameters

in	*REAL(dp) :: fSealed_area_↩ fraction*	sealed area fraction [1]
in	*REAL(dp) :: fast_interflow*	q_0 Fast runoff component [mm tst-1]
in	*REAL(dp) :: slow_interflow*	q_1 Slow runoff component [mm tst-1]
in	*REAL(dp) :: baseflow*	q_2 Baseflow [mm tsts-1]
in	*REAL(dp) :: direct_runoff*	q_D Direct runoff from impervious areas [mm tst-1]
out	*REAL(dp) :: total_runoff*	q_T Generated runoff [mm tst-1]

Authors

Vladyslav Prykhodko

Date

Dec 2012

Definition at line 252 of file mo_runoff.f90.

Referenced by mo_mhm::mhm(), mo_mhm_eval::mhm_eval(), mo_mrm_routing::mrm_routing(), mo_restart↩ ::read_restart_states(), mo_write_fluxes_states::updatedataset(), mo_init_states::variables_alloc(), mo_init_↩ states::variables_default_init(), and mo_restart::write_restart_files().

Here is the caller graph for this function:

15.127.2.2 subroutine, public mo_runoff::runoff_sat_zone (real(dp), intent(in) *k2,* real(dp), intent(inout) *sat_storage,* real(dp), intent(out) *baseflow*)

Runoff generation for the saturated zone.

Calculates the runoff generation for the saturated zone. If the level of the ground water reservoir is zero, then the baseflow is also zero. If the level of the ground water reservoir is greater than zero, then the baseflow is equal to baseflow recession coefficient times the level of the ground water reservoir, which will be then reduced by the value of baseflow.

Parameters

in	*REAL(dp) :: k2*	Baseflow recession coefficient [d-1]
in,out	*REAL(dp) :: sat_storage*	Groundwater storage [mm]
out	*REAL(dp) :: baseflow*	Baseflow [mm d-1]

Authors

Vladyslav Prykhodko

Date

Dec 2012

Definition at line 195 of file mo_runoff.f90.

Referenced by mo_mhm::mhm().

Here is the caller graph for this function:

15.127.2.3 subroutine, public mo_runoff::runoff_unsat_zone (real(dp), intent(in) *k1,* real(dp), intent(in) *kp,* real(dp), intent(in) *k0,* real(dp), intent(in) *alpha,* real(dp), intent(in) *karst_loss,* real(dp), intent(in) *pefec_soil,* real(dp), intent(in) *unsat_thresh,* real(dp), intent(inout) *sat_storage,* real(dp), intent(inout) *unsat_storage,* real(dp), intent(out) *slow_interflow,* real(dp), intent(out) *fast_interflow,* real(dp), intent(out) *perc*)

Runoff generation for the saturated zone.

Calculates the runoff generation for the unsaturated zone. Calculates percolation, interflow and baseflow. Updates upper soil and groundwater storages.

Parameters

in	*REAL(dp) :: k1*	Recession coefficient of the upper reservoir,lower outlet [d-1]
in	*REAL(dp) :: kp*	Percolation coefficient [d-1]
in	*REAL(dp) :: k0*	Recession coefficient of the upperreservoir, upper outlet [d-1]
in	*REAL(dp) :: alpha*	Exponent for the upper reservoir [-]
in	*REAL(dp) :: karst_loss*	Karstic percolation loss [-]
in	*REAL(dp) :: pefec_soil*	Input to the soil layer [mm]
in	*REAL(dp) :: unsat_thresh*	Threshold water depth in upper reservoir(for Runoff contribution) [mm]
in,out	*REAL(dp) :: sat_storage*	Groundwater storage [mm]
in,out	*REAL(dp) :: unsat_storage*	Upper soil storage [mm]
out	*REAL(dp) :: slow_interflow*	Slow runoff component [mm d-1]
out	*REAL(dp) :: fast_interflow*	Fast runoff component [mm d-1]
out	*REAL(dp) :: perc*	Percolation [mm d-1]

Authors

Vladyslav Prykhodko

Date

Dec 2012

Definition at line 78 of file mo_runoff.f90.

Referenced by mo_mhm::mhm().

Here is the caller graph for this function:

The documentation for this module was generated from the following file:

- mo_runoff.f90

15.128 mo_sce Module Reference

Shuffled Complex Evolution optimization algorithm.

Public Member Functions

- real(dp) function, dimension(size(pini, 1)), public sce (eval, functn, pini, prange, mymaxn, mymaxit, mykstop, mypcento, mypeps, myseed, myngs, mynpg, mynps, mynspl, mymings, myiniflg, myprint, mymask, myalpha, mybeta, tmp_file, popul_file, popul↩_file_append, parallel, restart, restart_file, bestf, neval, history)

  Shuffled Complex Evolution (SCE) algorithm for global optimization.
- subroutine comp (ngs2, npg, a, af, b, bf)
- subroutine sort_matrix (rb, ra)
- subroutine chkcst (x, bl, bu, mask, ibound)
- subroutine getpnt (idist, bl, bu, std, xi, mask, save_state, x)
- subroutine cce (s, sf, bl, bu, maskpara, xnstd, icall, maxn, maxit, save_state_gauss, functn, eval, alpha, beta, history, idot)

Private Member Functions

- subroutine parstt (x, bound, peps, mask, xnstd, gnrng, ipcnvg)

15.128.1 Detailed Description

Shuffled Complex Evolution optimization algorithm.

Optimization algorithm using Shuffled Complex Evolution strategy. Original version 2.1 of Qingyun Duan (1992) rewritten in Fortran 90.

Authors

Juliane Mai

Date

Feb 2013

Definition at line 11 of file mo_sce.f90.

15.128.2 Member Function/Subroutine Documentation

15.128.2.1 subroutine mo_sce::cce (real(dp), dimension(:, :), intent(inout) *s,* real(dp), dimension(:), intent(inout) *sf,* real(dp), dimension(:), intent(in) *bl,* real(dp), dimension(:), intent(in) *bu,* logical, dimension(:), intent(in) *maskpara,* real(dp), dimension(:), intent(in) *xnstd,* integer(i8), intent(inout) *icall,* integer(i8), intent(in) *maxn,* logical, intent(in) *maxit,* integer(i8), dimension(n_save_state), intent(inout) *save_state_gauss,* procedure(**objective_interface**), intent(in), pointer *functn,* procedure(**eval_interface**), intent(in), pointer *eval,* real(dp), intent(in) *alpha,* real(dp), intent(in) *beta,* real(dp), dimension(maxn), intent(inout) *history,* logical, intent(in) *idot*)

Definition at line 1747 of file mo_sce.f90.

References chkcst(), and getpnt().

Referenced by sce().

Here is the call graph for this function:

Here is the caller graph for this function:

15.128.2.2 subroutine mo_sce::chkcst (real(dp), dimension(:), intent(in) *x,* real(dp), dimension(:), intent(in) *bl,* real(dp), dimension(:), intent(in) *bu,* logical, dimension(:), intent(in) *mask,* integer(i4), intent(out) *ibound*)

Definition at line 1652 of file mo_sce.f90.

Referenced by cce(), and getpnt().

Here is the caller graph for this function:

15.128.2.3 subroutine mo_sce::comp (integer(i4), intent(in) *ngs2,* integer(i4), intent(in) *npg,* real(dp), dimension(:, :), intent(inout) *a,* real(dp), dimension(:), intent(inout) *af,* real(dp), dimension(size(a, 1), size(a, 2)), intent(out) *b,* real(dp), dimension(size(af)), intent(out) *bf*)

Definition at line 1564 of file mo_sce.f90.

Referenced by sce().

Here is the caller graph for this function:

15.128.2.4 subroutine mo_sce::getpnt (integer(i4), intent(in) *idist,* real(dp), dimension(:), intent(in) *bl,* real(dp), dimension(:), intent(in) *bu,* real(dp), dimension(:), intent(in) *std,* real(dp), dimension(:), intent(in) *xi,* logical, dimension(:), intent(in) *mask,* integer(i8), dimension(n_save_state), intent(inout) *save_state,* real(dp), dimension(size(xi, 1)), intent(out) *x*)

Definition at line 1693 of file mo_sce.f90.

References chkcst().

Referenced by cce(), and sce().

Here is the call graph for this function:

Here is the caller graph for this function:

15.128.2.5 **subroutine mo_sce::parstt (real(dp), dimension(:, :), intent(in) *x,* real(dp), dimension(:), intent(in) *bound,* real(dp), intent(in) *peps,* logical, dimension(:), intent(in) *mask,* real(dp), dimension(size(bound, 1)), intent(out) *xnstd,* real(dp), intent(out) *gnrng,* integer(i4), intent(out) *ipcnvg*)** `[private]`

Definition at line 1496 of file mo_sce.f90.

Referenced by sce().

Here is the caller graph for this function:

15.128.2.6 **real(dp) function, dimension(size(pini, 1)), public mo_sce::sce (procedure(eval_interface), intent(in), pointer *eval,* procedure(objective_interface), intent(in), pointer *functn,* real(dp), dimension(:), intent(in) *pini,* real(dp), dimension(:, :), intent(in) *prange,* integer(i8), intent(in), optional *mymaxn,* logical, intent(in), optional *mymaxit,* integer(i4), intent(in), optional *mykstop,* real(dp), intent(in), optional *mypcento,* real(dp), intent(in), optional *mypeps,* integer(i8), intent(in), optional *myseed,* integer(i4), intent(in), optional *myngs,* integer(i4), intent(in), optional *mynpg,* integer(i4), intent(in), optional *mynps,* integer(i4), intent(in), optional *mynspl,* integer(i4), intent(in), optional *mymings,* integer(i4), intent(in), optional *myiniflg,* integer(i4), intent(in), optional *myprint,* logical, dimension(size(pini, 1)), intent(in), optional *mymask,* real(dp), intent(in), optional *myalpha,* real(dp), intent(in), optional *mybeta,* character(len = ∗), intent(in), optional *tmp_file,* character(len = ∗), intent(in), optional *popul_file,* logical, intent(in), optional *popul_file_append,* logical, intent(in), optional *parallel,* logical, intent(in), optional *restart,* character(len = ∗), intent(in), optional *restart_file,* real(dp), intent(out), optional *bestf,* integer(i8), intent(out), optional *neval,* real(dp), dimension(:), intent(out), optional, allocatable *history*)**

Shuffled Complex Evolution (SCE) algorithm for global optimization.

Shuffled Complex Evolution method for global optimization
– version 2.1

by Qingyun Duan
Department of Hydrology & Water Resources
University of Arizona, Tucson, AZ 85721
(602) 621-9360, email: duan@hwr.arizona.edu

Written by Qingyun Duan, Oct 1990.
Revised by Qingyun Duan, Aug 1991.
Revised by Qingyun Duan, Apr 1992.

Re-written by Juliane Mai, Feb 2013.

Statement by Qingyun Duan:

This general purpose global optimization program is developed at the Department of Hydrology & Water Resources of the University of Arizona. Further information regarding the SCE-UA method can be obtained from Dr. Q. Duan, Dr. S. Sorooshian or Dr. V.K. Gupta at the address and phone number listed above. We request all users of this program make proper reference to the paper entitled 'Effective and Efficient Global Optimization for Conceptual Rainfall-runoff Models' by Duan, Q., S. Sorooshian, and V.K. Gupta, Water Resources Research, Vol 28(4), pp.1015-1031, 1992.

The function to be minimized is the first argument of DDS and must be defined as

```
function functn(p)
     use mo_kind, only: dp
     implicit none
     real(dp), dimension(:), intent(in) :: p
     real(dp) :: functn
end function functn
```

Parameters

in	*real(dp) :: functn(p)*	Function on which to search the optimum
in	*real(dp) :: pini(:)*	inital value of decision variables
in	*real(dp) :: prange(size(pini),2)*	Min/max range of decision variables
in	*integer(i8), optional :: mymaxn*	max no. of trials allowed before optimization is terminated DEFAULT: 1000_i8
in	*logical, optional :: mymaxit*	maximization (.true.) or minimization (.false.) of function DEFAULT: false
in	*integer(i4), optional :: mykstop*	number of shuffling loops in which the criterion value must change by given percentage before optimiz. is terminated DEFAULT: 10_i4
in	*real(dp), optional :: mypcento*	percentage by which the criterion value must change in given number of shuffling loops DEFAULT: 0.0001_dp
in	*real(dp), optional :: mypeps*	optimization is terminated if volume of complex has converged to given percentage of feasible space DEFAULT: 0.001_dp
in	*integer(i8), optional :: myseed*	initial random seed DEFAULT: get_timeseed
in	*integer(i4), optional :: myngs*	number of complexes in the initial population DEFAULT: 2_i4
in	*integer(i4), optional :: mynpg*	number of points in each complex DEFAULT: 2∗n+1

in	*integer(i4), optional :: mynps*	number of points in a sub-complex DEFAULT: n+1
in	*integer(i4), optional :: mynspl*	number of evolution steps allowed for each complex before complex shuffling DEFAULT: 2∗n+1
in	*integer(i4), optional :: mymings*	minimum number of complexes required, if the number of complexes is allowed to reduce as the optimization proceeds DEFAULT: ngs = number of complexes in initial population
in	*integer(i4), optional :: myiniflg*	flag on whether to include the initial point in population 0, not included 1, included (DEFAULT)
in	*integer(i4), optional :: myprint*	flag for controlling print-out after each shuffling loop 0, print information on the best point of the population 1, print information on every point of the population 2, no printing (DEFAULT) 3, same as 0 but print progress '.' on every function call 4, same as 1 but print progress '.' on every function call
in	*logical, optional :: mymask(size(pini))*	parameter included in optimization (true) or discarded (false) DEFAULT: .true.
in	*real(dp), optional :: myalpha*	parameter for reflection of points in complex DEFAULT: 0.8_dp
in	*real(dp), optional :: mybeta*	parameter for contraction of points in complex DEFAULT: 0.45_dp
in	*character(len=∗), optional :: tmp_file*	if given: write results after each evolution loop to temporal output file of that name **of headlines: 7**

format: '# nloop icall ngs1 bestf worstf ...
... gnrng (bestx(j),j=1,nn)'

Parameters

in	*character(len=∗), optional :: popul_file*	if given: write whole population to file of that name **of headlines: 1**

format: #_evolution_loop, xf(i), (x(i,j),j=1,nn)
total number of lines written <= neval <= mymaxn

Parameters

in	*logical, optional :: popul_file_↩ append*	if true, append to existing population file (default: false)
in	*logical, optional :: parallel*	sce runs in parallel (true) or not (false) parallel sce should only be used if model/ objective is not parallel DEFAULT: .false.
in	*logical, optional :: restart*	if .true.: restart former sce run from restart_file
in	*character(len=∗), optional :: restart_file*	file name for read/write of restart file (default: mo_sce.restart)

out	*real(dp), optional :: bestf*	the best value of the function.
out	*integer(i8), optional :: neval*	number of function evaluations needed.
out	*real(dp), optional, allocatable :: history(:)*	the history of best function values, history(neval)=bestf

Returns

real(dp) :: bestx(size(pini)) — The parameters of the point which is estimated to minimize/maximize the function.

Note

Maximal number of parameters is 1000.
SCE is OpenMP enabled on the loop over the complexes.
OMP_NUM_THREADS $>$ 1 does not give reproducible results even when seeded!

Author

Juliane Mai

Date

Feb 2013

Definition at line 229 of file mo_sce.f90.

References cce(), comp(), mo_string_utils::compress(), getpnt(), parstt(), set_optional(), sort_matrix(), write_best↩_final(), write_best_intermediate(), write_population(), and write_termination_case().

Here is the call graph for this function:

15.128.2.7 subroutine mo_sce::sort_matrix (real(dp), dimension(:, :), intent(inout) *rb*, real(dp), dimension(:), intent(inout) *ra*)

Definition at line 1620 of file mo_sce.f90.

Referenced by sce().

Here is the caller graph for this function:

The documentation for this module was generated from the following file:

- mo_sce.f90

15.129 mo_set_netcdf_outputs Module Reference

Defines the structure of the netCDF to write the output in.

Public Member Functions

- subroutine, public set_netcdf (NoNetcdfVars, nrows, ncols)

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

15.129.1 Detailed Description

Defines the structure of the netCDF to write the output in.

All output variables are initialized for the NetCDF.

Authors

Matthias Zink

Date

Apr 2013

Definition at line 13 of file mo_set_netcdf_outputs.f90.

15.129.2 Member Function/Subroutine Documentation

15.129.2.1 subroutine, public mo_set_netcdf_outputs::set_netcdf (integer(i4), intent(in) *NoNetcdfVars,* integer(i4), intent(in) *nrows,* integer(i4), intent(in) *ncols*)

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

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

Parameters

in	*integer(i4) :: NoNetcdfVars*	
in	*integer(i4) :: nrows*	
in	*integer(i4) :: ncols*	

Authors

Matthias Zink

Date

Apr 2013

Definition at line 55 of file mo_set_netcdf_outputs.f90.

The documentation for this module was generated from the following file:

- mo_set_netcdf_outputs.f90

15.130 mo_snow_accum_melt Module Reference

Snow melting and accumulation.

Public Member Functions

- 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.

15.130.1 Detailed Description

Snow melting and accumulation.

This module calculates snow melting and accumulation.

Authors

Vladyslav Prykhodko

Date

Dec 2012

Definition at line 13 of file mo_snow_accum_melt.f90.

15.130.2 Member Function/Subroutine Documentation

15.130.2.1 subroutine, public mo_snow_accum_melt::snow_accum_melt (real(dp), intent(in) *deg_day_incr,* real(dp), intent(in) *deg_day_max,* real(dp), intent(in) *deg_day_noprec,* real(dp), intent(in) *prec,* real(dp), intent(in) *temperature,* real(dp), intent(in) *temperature_thresh,* real(dp), intent(in) *thrfall,* real(dp), intent(inout) *snow_pack,* real(dp), intent(out) *deg_day,* real(dp), intent(out) *melt,* real(dp), intent(out) *prec_effect,* real(dp), intent(out) *rain,* real(dp), intent(out) *snow*)

Snow melting and accumulation.

Separates throughfall into rain and snow by comparing the temperature with the treshhold. by comparing the temperature with the treshhold. Calculates degree daily factor. Calculates snow melting rates. Calculates snow, rain and effective precipitation depth and snow pack.

Parameters

in	*REAL(dp) :: deg_day_incr*	Increase of the Degree-day factor per mm of increasein precipitation [s-1 degreeC-1]
in	*REAL(dp) :: deg_day_max*	Maximum Degree-day factor [m-1 degreeC-1]
in	*REAL(dp) :: deg_day_noprec*	Degree-day factor with no precipitation [m-1 degreeC-1]
in	*REAL(dp) :: prec*	Daily mean precipitation [m]
in	*REAL(dp) :: temperature*	Daily mean temperature [degreeC]
in	*REAL(dp) :: temperature_thresh*	Threshold temperature for snow/rain [degreeC]
in	*REAL(dp) :: thrfall*	Throughfall [m s-1]

in,out	*REAL(dp) :: snow_pack*	Snow pack [m]
out	*REAL(dp) :: deg_day*	Degree-day factor [m s-1 degreeC-1]
out	*REAL(dp) :: melt*	Melting snow depth [m s-1]
out	*REAL(dp) :: prec_effect*	Effective precipitation depth (snow melt + rain) [m]
out	*REAL(dp) :: rain*	Rain precipitation depth [m]
out	*REAL(dp) :: snow*	Snow precipitation depth [m]

Authors

Vladyslav Prykhodko

Date

Dec 2012

Definition at line 71 of file mo_snow_accum_melt.f90.

Referenced by mo_mhm::mhm().

Here is the caller graph for this function:

The documentation for this module was generated from the following file:

- mo_snow_accum_melt.f90

15.131 mo_soil_moisture Module Reference

Soil moisture of the different layers.

Public Member Functions

- 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.

Private Member Functions

- elemental pure real(dp)
  function, private feddes_et_reduction (soil_moist, soil_moist_FC, wilting_point, frac_roots)

  stress factor for reducing evapotranspiration based on actual soil moisture
- elemental pure real(dp)
  function, private jarvis_et_reduction (soil_moist, soil_moist_sat, wilting_point, frac_roots, jarvis_thresh_c1)

  stress factor for reducing evapotranspiration based on actual soil moisture

15.131.1 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

Definition at line 20 of file mo_soil_moisture.f90.

15.131.2 Member Function/Subroutine Documentation

15.131.2.1 elemental pure real(dp) function, private mo_soil_moisture::feddes_et_reduction (real(dp), intent(in) *soil_moist,* real(dp), intent(in) *soil_moist_FC,* real(dp), intent(in) *wilting_point,* real(dp), intent(in) *frac_roots*) [private]

stress factor for reducing evapotranspiration based on actual soil moisture

Potential evapotranspiration is reduced to 0 if SM is lower PWP. PET is equal fraction of roots if soil moisture is exceeding field capacity. If soil moisture is in between PWP and FC PET is reduced by fraction of roots times a stress factor. The ET reduction factor f is estimated as

$$f = \begin{cases} f_{roots} & if\theta \geq \theta_{fc} \\ f_{roots} \cdot \frac{\theta-\theta_{pwp}}{\theta_{fc}-\theta_{pwp}} & if\theta < \theta_{fc} \\ 0 & if\theta < \theta_{pwp} \end{cases}$$

Parameters

in	*real(dp) :: soil_moist*	Soil moisture of each horizon [mm]
in	*real(dp) :: soil_moist_FC*	Soil moisture below which actual ET is reduced [mm]
in	*real(dp) :: wilting_point*	Permanent wilting point
in	*real(dp) :: frac_roots*	Fraction of Roots in soil horizon is reduced [mm]

Returns

real(dp) :: feddes_et_reduction; et reduction factor

Authors

Matthias Cuntz, Cueneyd Demirel, Matthias Zink

Date

March 2017

Definition at line 332 of file mo_soil_moisture.f90.

Referenced by soil_moisture().

Here is the caller graph for this function:

15.131.2.2 elemental pure real(dp) function, private mo_soil_moisture::jarvis_et_reduction (real(dp), intent(in) *soil_moist,* real(dp), intent(in) *soil_moist_sat,* real(dp), intent(in) *wilting_point,* real(dp), intent(in) *frac_roots,* real(dp), intent(in) *jarvis_thresh_c1*) [private]

stress factor for reducing evapotranspiration based on actual soil moisture

The soil moisture stress factor is estimated based on the normalized soil water content. The normalized soil water content θ_{norm} is estimated as:

$$\theta_{norm} = \frac{\theta - \theta_{pwp}}{\theta_{sat} - \theta_{pwp}}$$

The ET reduction factor f is estimated as

$$f = \begin{cases} f_{roots} & if\,\theta_{norm} \geq jarvis_sm_threshold_c1 \\ f_{roots}\frac{\theta_{norm}}{jarvis_sm_threshold_c1} & if\,\theta_{norm} < jarvis_sm_threshold_c1 \end{cases}$$

Parameters

in	*real(dp) :: soil_moist*	Soil moisture of each horizon [mm]
in	*real(dp) :: soil_moist_sat*	saturated Soil moisture content [mm]
in	*real(dp) :: wilting_point*	Permanent wilting point
in	*real(dp) :: frac_roots*	Fraction of Roots in soil horizon is reduced [mm]
in	*real(dp) :: jarvis_thresh_c1*	parameter C1 from Jarvis formulation

Returns

real(dp) :: jarvis_et_reduction; et reduction factor

Authors

Cueneyd Demirel, Matthias Zink

Date

March 2017

Definition at line 405 of file mo_soil_moisture.f90.

Referenced by soil_moisture().

Here is the caller graph for this function:

15.131.2.3 subroutine, public mo_soil_moisture::soil_moisture (integer(i4), intent(in) *processCase,* real(dp), intent(in) *frac_sealed,* real(dp), intent(in) *water_thresh_sealed,* real(dp), intent(in) *pet,* real(dp), intent(in) *evap_coeff,* real(dp), dimension(:), intent(in) *soil_moist_sat,* real(dp), dimension(:), intent(in) *frac_roots,* real(dp), dimension(:), intent(in) *soil_moist_FC,* real(dp), dimension(:), intent(in) *wilting_point,* real(dp), dimension(:), intent(in) *soil_moist_exponen,* real(dp), intent(in) *jarvis_thresh_c1,* real(dp), intent(in) *aet_canopy,* real(dp), intent(inout) *prec_effec,* real(dp), intent(inout) *runoff_sealed,* real(dp), intent(inout) *storage_sealed,* real(dp), dimension(size(soil_moist_sat, 1)), intent(inout) *infiltration,* real(dp), dimension(size(soil_moist_sat, 1)), intent(inout) *soil_moist,* real(dp), dimension(size(soil_moist_sat, 1)), intent(out) *aet,* real(dp), intent(out) *aet_sealed*)

Soil moisture in different soil horizons.

Infiltration I from one layer $k-1$ to the next k on pervious areas is calculated as (omit t)

$$I[k] = I[k-1](\theta[k]/\theta_{sat}[k])^{\beta[k]}$$

Then soil moisture can be calculated as (omit k)

$$\theta[t] = \theta[t-1] + I[t] - ET[t]$$

with ET (omit $[k,t]$) being

$$ET = f_{\text{roots}} \cdot f_{SM} \cdot PET$$

.

Parameters

in	*integer(i4) :: processCase*	1 - Feddes equation for PET reduction2 - Jarvis equation for PET reduction3 - Jarvis equation for PET reduction and FC dependency on root fraction coefficient
in	*real(dp) :: frac_sealed*	Fraction of sealed area
in	*real(dp) :: water_thresh_↩ sealed*	Threshhold water depth in impervious areas [mm/s]
in	*real(dp) :: pet*	Reference evapotranspiration [mm/s]
in	*real(dp) :: evap_coeff*	Evaporation coefficent for free-water surface of that current month
in	*real(dp), dimension(:) :: soil_moist_sat*	Saturation soil moisture for each horizon [mm]

in	*real(dp), dimension(:) :: frac_roots*	Fraction of Roots in soil horizon
in	*real(dp), dimension(:) :: soil_moist_FC*	Soil moisture below which actual ET is reduced [mm]
in	*real(dp), dimension(:) :: wilting_point*	Permanent wilting point for each horizon [mm]
in	*real(dp), dimension(:) :: soil_moist_↩ exponen*	Exponential parameter to how non-linear is the soil water retention
in	*real(dp) :: jarvis_thresh_c1*	Jarvis critical value for normalized soil water content
in	*real(dp) :: aet_canopy*	Actual ET from canopy [mm/s]
in,out	*real(dp) :: prec_effec*	Effective precipitation (rain + snow melt) [mm]
in,out	*real(dp) :: runoff_sealed*	Direct runoff from impervious areas
in,out	*real(dp) :: storage_sealed*	Retention storage of impervious areas
in,out	*real(dp), dimension(size(soil↩ _moist_sat, 1)) :: infiltration*	Recharge, infiltration intensity oreffective precipitation of each horizon [mm/s]
in,out	*real(dp), dimension(size(soil↩ _moist_sat, 1)) :: soil_moist*	Soil moisture of each horizon [mm]
out	*real(dp), dimension(size(soil↩ _moist_sat, 1)) :: aet*	actual ET [mm/s]
out	*real(dp) :: aet_sealed*	actual ET from free-water surfaces,i.e impervious cover [mm/s]

Authors

Matthias Cuntz

Date

Dec 2012

Definition at line 94 of file mo_soil_moisture.f90.

References 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:

The documentation for this module was generated from the following file:

- mo_soil_moisture.f90

15.132 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.

Public Member Functions

- 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)

Private Member Functions

- subroutine spatial_aggregation_3d (data2, cellsize2, cellsize1, mask1, mask2, data1)

15.132.1 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

Definition at line 15 of file mo_spatial_agg_disagg_forcing.f90.

15.132.2 Member Function/Subroutine Documentation

15.132.2.1 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.

15.132.2.2 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.

15.132.2.3 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.

15.132.2.4 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.

The documentation for this module was generated from the following file:

- mo_spatial_agg_disagg_forcing.f90

15.133 mo_spatialsimilarity Module Reference

Routines for bias insensitive comparison of spatial patterns.

Data Types

- interface nndv

  Calculates the number of neighboring dominating values, a measure for spatial dissimilarity.
- interface pd

  Calculates pattern dissimilarity (PD) measure.

Public Member Functions

- real(sp) function nndv_sp (mat1, mat2, mask, valid)
- real(dp) function nndv_dp (mat1, mat2, mask, valid)
- real(sp) function pd_sp (mat1, mat2, mask, valid)
- real(dp) function pd_dp (mat1, mat2, mask, valid)

15.133.1 Detailed Description

Routines for bias insensitive comparison of spatial patterns.

These routines are based on the idea that spatial similarity can be assessed by comparing the magnitude of neighboring pixels (e.g. is the neighboring pixel larger or smaller).

Author

Matthias Zink

Date

Mar 2013

Definition at line 11 of file mo_spatialsimilarity.f90.

15.133.2 Member Function/Subroutine Documentation

15.133.2.1 real(dp) function mo_spatialsimilarity::nndv_dp (real(dp), dimension(:, :), intent(in) *mat1,* real(dp), dimension(:, :), intent(in) *mat2,* logical, dimension(:, :), intent(in), optional *mask,* logical, intent(out), optional *valid*)

Definition at line 336 of file mo_spatialsimilarity.f90.

References mo_spatialsimilarity::nndv::nndv_dp().

Here is the call graph for this function:

15.133.2.2 real(sp) function mo_spatialsimilarity::nndv_sp (real(sp), dimension(:, :), intent(in) *mat1,* real(sp), dimension(:, :), intent(in) *mat2,* logical, dimension(:, :), intent(in), optional *mask,* logical, intent(out), optional *valid*)

Definition at line 250 of file mo_spatialsimilarity.f90.

References mo_spatialsimilarity::nndv::nndv_sp().

Here is the call graph for this function:

15.133.2.3 real(dp) function mo_spatialsimilarity::pd_dp (real(dp), dimension(:, :), intent(in) *mat1,* real(dp), dimension(:, :), intent(in) *mat2,* logical, dimension(:, :), intent(in), optional *mask,* logical, intent(out), optional *valid*)

Definition at line 515 of file mo_spatialsimilarity.f90.

References mo_spatialsimilarity::pd::pd_dp().

Here is the call graph for this function:

15.133.2.4 real(sp) function mo_spatialsimilarity::pd_sp (real(sp), dimension(:, :), intent(in) *mat1,* real(sp), dimension(:, :), intent(in) *mat2,* logical, dimension(:, :), intent(in), optional *mask,* logical, intent(out), optional *valid*)

Definition at line 424 of file mo_spatialsimilarity.f90.

References mo_spatialsimilarity::pd::pd_sp().

Here is the call graph for this function:

The documentation for this module was generated from the following file:

- mo_spatialsimilarity.f90

15.134 mo_standard_score Module Reference

Routines for calculating the normalization (anomaly)/standard score/z score and the deseasonalized (standard score on monthly basis) values of a time series.

Data Types

- interface classified_standard_score

  Calculates the classified standard score (e.g. classes are months).
- interface standard_score

  Calculates the standard score / normalization (anomaly) / z-score.

Public Member Functions

- real(dp) function, dimension(size(data, dim=1)) standard_score_dp (data, mask)
- real(sp) function, dimension(size(data, dim=1)) classified_standard_score_sp (data, classes, mask)
- real(dp) function, dimension(size(data, dim=1)) classified_standard_score_dp (data, classes, mask)

Private Member Functions

- real(sp) function, dimension(size(data, dim=1)) standard_score_sp (data, mask)

15.134.1 Detailed Description

Routines for calculating the normalization (anomaly)/standard score/z score and the deseasonalized (standard score on monthly basis) values of a time series.

In environmental research often the centralization and standardization are estimated for characterizing the dynamics of a signal.

Author

Matthias Zink

Date

May 2015

Definition at line 12 of file mo_standard_score.f90.

15.134.2 Member Function/Subroutine Documentation

15.134.2.1 real(dp) function, dimension(size(data, dim = 1)) mo_standard_score::classified_standard_score_dp (real(dp), dimension(:), intent(in) *data,* integer, dimension(:), intent(in) *classes,* logical, dimension(:), intent(in), optional *mask*)

Definition at line 321 of file mo_standard_score.f90.

References mo_standard_score::classified_standard_score::classified_standard_score_dp().

Here is the call graph for this function:

15.134.2.2 real(sp) function, dimension(size(data, dim = 1)) mo_standard_score::classified_standard_score_sp (real(sp), dimension(:), intent(in) *data,* integer, dimension(:), intent(in) *classes,* logical, dimension(:), intent(in), optional *mask*)

Definition at line 263 of file mo_standard_score.f90.

References mo_standard_score::classified_standard_score::classified_standard_score_sp().

Here is the call graph for this function:

15.134.2.3 real(dp) function, dimension(size(data, dim = 1)) mo_standard_score::standard_score_dp (real(dp), dimension(:), intent(in) *data*, logical, dimension(:), intent(in), optional *mask*)

Definition at line 233 of file mo_standard_score.f90.

References mo_standard_score::standard_score::standard_score_dp().

Here is the call graph for this function:

15.134.2.4 real(sp) function, dimension(size(data, dim = 1)) mo_standard_score::standard_score_sp (real(sp), dimension(:), intent(in) *data*, logical, dimension(:), intent(in), optional *mask*) [private]

Definition at line 204 of file mo_standard_score.f90.

References mo_standard_score::standard_score::standard_score_sp().

Here is the call graph for this function:

The documentation for this module was generated from the following file:

- mo_standard_score.f90

15.135 mo_startup Module Reference

Startup procedures for mHM.

Public Member Functions

- subroutine, public mhm_initialize

  Initialize main mHM variables.

- subroutine constants_init

  Initialize mHM constants.

- subroutine l2_variable_init (iBasin, level0_iBasin, level2_iBasin)

  Initalize Level-2 meteorological forcings data.

15.135.1 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

Definition at line 15 of file mo_startup.f90.

15.135.2 Member Function/Subroutine Documentation

15.135.2.1 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_message::message(), and mo_neutrons::tabularintegralafast().

Referenced by mhm_initialize().

Here is the call graph for this function:

Here is the caller graph for this function:

15.135.2.2 subroutine mo_startup::l2_variable_init (integer(i4), intent(in) *iBasin,* type(grid), intent(in) *level0_iBasin,* type(grid), intent(inout) *level2_iBasin*)

Initalize Level-2 meteorological forcings data.

following tasks are performed 1) cell id & numbering 2) mask creation 3) append variable of intrest to global ones

Parameters

in	*integer(i4) :: iBasin*	Basin Id
in	*type(Grid) :: level0_iBasin*	
in,out	*type(Grid) :: level2_iBasin*	

Authors

Rohini Kumar

Date

Feb 2013

Definition at line 211 of file mo_startup.f90.

References mo_grid::init_lowres_level(), mo_message::message(), and mo_read_spatial_data::read_header_↩ ascii().

Referenced by mhm_initialize().

Here is the call graph for this function:

Here is the caller graph for this function:

15.135.2.3 subroutine, public mo_startup::mhm_initialize ( )

Initialize main mHM variables.

Initialize main mHM variables for a given basin. Calls the following procedures in this order:

- Constant initialization.
- Generate soil database.
- Checking inconsistencies input fields.
- Variable initialization at level-0.
- Variable initialization at level-1.
- Variable initialization at level-11.
- Space allocation of remaining variable/parameters. Global variables will be used at this stage.

Authors

Luis Samaniego, Rohini Kumar

Date

Dec 2012

Definition at line 70 of file mo_startup.f90.

References constants_init(), l2_variable_init(), mo_common_restart::read_grid_info(), mo_grid::set_basin_↩indices(), and mo_init_states::variables_alloc().

Referenced by mhm_driver().

Here is the call graph for this function:

Here is the caller graph for this function:

The documentation for this module was generated from the following file:

- mo_startup.f90

15.136 mo_string_utils Module Reference

String utilities.

Collaboration diagram for mo_string_utils:

Data Types

- interface num2str

  Convert to string.
- interface numarray2str

  Convert to string.

Public Member Functions

- character(len(whitespaces))
  function, public compress (whiteSpaces, n)
- subroutine, public divide_string (string, delim, strArr)

  Divide string in substrings.
- logical function, public equalstrings (string1, string2)

- logical function, public nonull (str)

  Checks if string was already used.
- character(len=256) function,
  dimension(:), allocatable,
  public splitstring (string, delim)
- logical function, public startswith (string, start)
- character(len=len_trim(upper))
  function, public tolower (upper)

  Convert to lower case.
- character(len=len_trim(lower))
  function, public toupper (lower)
- integer(i4) function,
  dimension(:), allocatable,
  public str2num (string)

Public Attributes

- character(len=∗), parameter, public separator = repeat('-', 70)

Private Member Functions

- pure character(len=10) function i42str (nn, form)
- pure character(len=20) function i82str (nn, form)
- pure character(len=32) function sp2str (rr, form)
- pure character(len=32) function dp2str (rr, form)
- pure character(len=10) function log2str (ll, form)
- character(len=size(arr)) function i4array2str (arr)

15.136.1 Detailed Description

String utilities.

This module provides string conversion and checking utilities.

Authors

Matthias Cuntz, Matthias Zink, Giovanni Dalmasso, David Schaefer

Date

Dec 2011

Definition at line 10 of file mo_string_utils.f90.

15.136.2 Member Function/Subroutine Documentation

15.136.2.1 character(len(whitespaces)) function, public mo_string_utils::compress (character(len = ∗), intent(in) *whiteSpaces*, integer(i4), intent(out), optional *n*)

Definition at line 237 of file mo_string_utils.f90.

Referenced by mo_sce::sce().

Here is the caller graph for this function:

15.136.2.2 subroutine, public mo_string_utils::divide_string (character(len = ∗), intent(in) *string,* character(len = ∗), intent(in) *delim,* character(len = ∗), dimension(:), intent(out), allocatable *strArr*)

Divide string in substrings.

Divides a string in several substrings (array of strings) with the help of a user specified delimiter.

Parameters

in	*CHARACTER(len=∗), INTENT(IN) :: string*	- string to be divided
in	*CHARACTER(len=∗), INTENT(IN) :: delim*	- delimiter specifying places for division
out	*CHARACTER(len=∗), DIMENSION(:), ALLOCATABLE, INTENT(OUT) :: strArr*	Array of substrings, has to be allocateable and is handed to the routine unallocated

Author

Matthias Zink

Date

Oct 2012

Definition at line 331 of file mo_string_utils.f90.

Referenced by mo_read_forcing_nc::get_time_vector_and_select().

Here is the caller graph for this function:

15.136.2.3 pure character(len = 32) function mo_string_utils::dp2str (real(dp), intent(in) *rr*, character(len = ∗), intent(in), optional *form*) [private]

Definition at line 847 of file mo_string_utils.f90.

References mo_string_utils::num2str::dp2str().

Here is the call graph for this function:

15.136.2.4 logical function, public mo_string_utils::equalstrings (character(len = ∗), intent(in) *string1*, character(len = ∗), intent(in) *string2*)

Definition at line 425 of file mo_string_utils.f90.

References str2num().

Here is the call graph for this function:

15.136.2.5 pure character(len = 10) function mo_string_utils::i42str (integer(i4), intent(in) *nn*, character(len = ∗), intent(in), optional *form*) [private]

Definition at line 796 of file mo_string_utils.f90.

References mo_string_utils::num2str::i42str().

Here is the call graph for this function:

15.136.2.6 character(len = size(arr)) function mo_string_utils::i4array2str (integer(i4), dimension(:), intent(in) *arr*) [private]

Definition at line 880 of file mo_string_utils.f90.

15.136.2.7 pure character(len = 20) function mo_string_utils::i82str (integer(i8), intent(in) *nn*, character(len = ∗), intent(in), optional *form*) [private]

Definition at line 813 of file mo_string_utils.f90.

References mo_string_utils::num2str::i82str().

Here is the call graph for this function:

15.136.2.8 pure character(len = 10) function mo_string_utils::log2str (logical, intent(in) *ll*, character(len = ∗), intent(in), optional *form*) [private]

Definition at line 864 of file mo_string_utils.f90.

References mo_string_utils::num2str::log2str().

Here is the call graph for this function:

15.136.2.9 logical function, public mo_string_utils::nonull (character(len = ∗), intent(in) *str*)

Checks if string was already used.

Checks if string was already used, i.e. does not contain NULL character anymore.

Parameters

in	*character(len=∗) :: str*	String

Returns

logical :: used — .true.: string was already set; .false.: string still in initialised state

Author

Matthias Cuntz

Date

Jan 2012

Definition at line 497 of file mo_string_utils.f90.

Referenced by mo_ncwrite::create_netcdf().

Here is the caller graph for this function:

15.136.2.10 pure character(len = 32) function mo_string_utils::sp2str (real(sp), intent(in) *rr,* character(len = ∗), intent(in), optional *form*) [private]

Definition at line 830 of file mo_string_utils.f90.

References mo_string_utils::num2str::sp2str().

Here is the call graph for this function:

15.136.2.11 character(len = 256) function, dimension(:), allocatable, public mo_string_utils::splitstring (character(len = ∗), intent(in) *string,* character(len = ∗), intent(in) *delim*)

Definition at line 560 of file mo_string_utils.f90.

References str2num().

Here is the call graph for this function:

15.136.2.12 logical function, public mo_string_utils::startswith (character(len = ∗), intent(in) *string,* character(len = ∗), intent(in) *start*)

Definition at line 633 of file mo_string_utils.f90.

References str2num().

Here is the call graph for this function:

15.136.2.13 integer(i4) function, dimension(:), allocatable, public mo_string_utils::str2num (character(len = ∗), intent(in) *string*)

Definition at line 940 of file mo_string_utils.f90.

Referenced by equalstrings(), splitstring(), and startswith().

Here is the caller graph for this function:

15.136.2.14 character(len = len_trim(upper)) function, public mo_string_utils::tolower (character(len = ∗), intent(in) *upper*)

Convert to lower case.

Convert all upper case letters in string to lower case letters.

Parameters

in	*character(len=∗) :: upper*	String

Returns

character(len=len_trim(upper)) :: low — String where all uppercase in input is converted to lowercase

Author

Matthias Cuntz - modified from Echam5, (C) MPI-MET, Hamburg, Germany

Date

Dec 2011

Definition at line 699 of file mo_string_utils.f90.

Referenced by mo_nml::position_nml().

Here is the caller graph for this function:

15.136.2.15 character(len = len_trim(lower)) function, public mo_string_utils::toupper (character(len = ∗), intent(in) *lower*)

Definition at line 769 of file mo_string_utils.f90.

Referenced by mo_utils::special_value_dp(), and mo_utils::special_value_sp().

Here is the caller graph for this function:

15.136.3 Member Data Documentation

15.136.3.1 character(len = ∗), parameter, public mo_string_utils::separator = repeat('-', 70)

Definition at line 182 of file mo_string_utils.f90.

The documentation for this module was generated from the following file:

- mo_string_utils.f90

15.137 mo_template Module Reference

Template for future module developments.

Collaboration diagram for mo_template:

Data Types

- interface mean

  The average.

Public Member Functions

- elemental pure real(dp) function, public circum (radius)

  Circumference of a circle.

Public Attributes

- 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.

Private Member Functions

- real(dp) function mean_dp (dat, mask)
- real(sp) function mean_sp (dat, mask)

Private Attributes

- integer(i4), parameter itest = 1

15.137.1 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

Definition at line 15 of file mo_template.f90.

15.137.2 Member Function/Subroutine Documentation

15.137.2.1 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.

15.137.2.2 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.

References mo_template::mean::mean_dp().

Here is the call graph for this function:

15.137.2.3 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.

References mo_template::mean::mean_sp().

Here is the call graph for this function:

15.137.3 Member Data Documentation

15.137.3.1 integer(i4), parameter mo_template::itest = 1 [private]

Definition at line 92 of file mo_template.f90.

15.137.3.2 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.

15.137.3.3 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.

The documentation for this module was generated from the following file:

- mo_template.f90

15.138 mo_temporal_aggregation Module Reference

Temporal aggregation for time series (averaging)

Data Types

- interface day2mon_average

  Day-to-month average (day2mon_average)
- interface hour2day_average

  Hour-to-day average (hour2day_average)

Private Member Functions

- subroutine day2mon_average_dp (daily_data, yearS, monthS, dayS, mon_avg, misval, rm_misval)
- subroutine hour2day_average_dp (hourly_data, yearS, monthS, dayS, hourS, day_avg, misval, rm_misval)

15.138.1 Detailed Description

Temporal aggregation for time series (averaging)

This module does temporal aggregation (averaging) of time series

Authors

Oldrich Rakovec, Rohini Kumar

Date

October 2015

Definition at line 10 of file mo_temporal_aggregation.f90.

15.138.2 Member Function/Subroutine Documentation

15.138.2.1 subroutine mo_temporal_aggregation::day2mon_average_dp (real(dp), dimension(:), intent(in) *daily_data,* integer(i4), intent(in) *yearS,* integer(i4), intent(in) *monthS,* integer(i4), intent(in) *dayS,* real(dp), dimension(:), intent(inout), allocatable *mon_avg,* real(dp), intent(in), optional *misval,* logical, intent(in), optional *rm_misval*) [private]

Definition at line 166 of file mo_temporal_aggregation.f90.

References mo_julian::dec2date(), mo_julian::julday(), and mo_julian::ndays().

Here is the call graph for this function:

15.138.2.2 subroutine mo_temporal_aggregation::hour2day_average_dp (real(dp), dimension(:), intent(in) *hourly_data,* integer(i4), intent(in) *yearS,* integer(i4), intent(in) *monthS,* integer(i4), intent(in) *dayS,* integer(i4), intent(in) *hourS,* real(dp), dimension(:), intent(inout), allocatable *day_avg,* real(dp), intent(in), optional *misval,* logical, intent(in), optional *rm_misval*) [private]

Definition at line 273 of file mo_temporal_aggregation.f90.

References mo_julian::dec2date(), and mo_julian::julday().

Here is the call graph for this function:

The documentation for this module was generated from the following file:

- mo_temporal_aggregation.f90

15.139 mo_temporal_disagg_forcing Module Reference

Temporal disaggregation of daily input values.

Public Member Functions

- 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.

15.139.1 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

Definition at line 14 of file mo_temporal_disagg_forcing.f90.

15.139.2 Member Function/Subroutine Documentation

15.139.2.1 elemental pure subroutine, public mo_temporal_disagg_forcing::temporal_disagg_forcing (logical, intent(in) *isday,* real(dp), intent(in) *ntimesteps_day,* real(dp), intent(in) *prec_day,* real(dp), intent(in) *pet_day,* real(dp), intent(in) *temp_day,* real(dp), intent(in) *fday_prec,* real(dp), intent(in) *fday_pet,* real(dp), intent(in) *fday_temp,* real(dp), intent(in) *fnight_prec,* real(dp), intent(in) *fnight_pet,* real(dp), intent(in) *fnight_temp,* real(dp), intent(in) *temp_weights,* real(dp), intent(in) *pet_weights,* real(dp), intent(in) *pre_weights,* logical, intent(in) *read_meteo_weights,* real(dp), intent(out) *prec,* real(dp), intent(out) *pet,* real(dp), intent(out) *temp*)

Temporally distribute daily mean forcings onto time step.

Calculates actual precipitation, PET and temperature from daily mean inputs. Precipitation and PET are distributed with predefined factors onto the day. Temperature gets a predefined amplitude added on day and substracted at night. Alternatively, weights for each hour and month can be given and disaggregation is using these as factors for PET and temperature. Precipitation is distributed uniformly.

Parameters

in	*logical :: isday*	is day or night
in	*real(dp) :: ntimesteps_day*	# of time steps per day
in	*real(dp) :: prec_day*	Daily mean precipitation [mm/s]
in	*real(dp) :: pet_day*	Daily mean ET [mm/s]
in	*real(dp) :: temp_day*	Daily mean air temperature [K]
in	*real(dp) :: fday_prec*	Daytime fraction of precipitation
in	*real(dp) :: fday_pet*	Daytime fraction of PET
in	*real(dp) :: fday_temp*	Daytime air temparture increase
in	*real(dp) :: fnight_prec*	Daytime fraction of precipitation
in	*real(dp) :: fnight_pet*	Daytime fraction of PET
in	*real(dp) :: fnight_temp*	Daytime air temparture increase
in	*real(dp) :: temp_weights*	weights for average temperature
in	*real(dp) :: pet_weights*	weights for PET
in	*real(dp) :: pre_weights*	weights for precipitation
in	*logical :: read_meteo_weights*	flag indicating that weights should be used
out	*real(dp) :: prec*	Actual precipitation [mm/s]
out	*real(dp) :: pet*	Reference ET [mm/s]
out	*real(dp) :: temp*	Air temperature [K]

Authors

Matthias Cuntz

Date

Dec 2012

Definition at line 73 of file mo_temporal_disagg_forcing.f90.

Referenced by mo_mhm::mhm().

Here is the caller graph for this function:

The documentation for this module was generated from the following file:

- mo_temporal_disagg_forcing.f90

15.140 mo_timer Module Reference

Timing routines.

Collaboration diagram for mo_timer:

Public Member Functions

- subroutine, public timer_check (timer)

  Check a timer.
- subroutine, public timer_clear (timer)

  Reset a timer.
- real(sp) function, public timer_get (timer)

  Return a timer.
- subroutine, public timer_print (timer)

  Print a timer.

- subroutine, public timer_start (timer)

  Start a timer.
- subroutine, public timer_stop (timer)

  Stop a timer.
- subroutine, public timers_init

  Initialise timer module.

Public Attributes

- integer(i4), parameter, public max_timers = 99

  max number of timers allowed
- integer(i4), save, public cycles_max

  max value of clock allowed by system
- real(sp), save, public clock_rate

  clock_rate in seconds for each cycle
- integer(i4), dimension(max_timers), save, public cycles1

  cycle number at start for each timer
- integer(i4), dimension(max_timers), save, public cycles2

  cycle number at stop for each timer
- real(sp), dimension(max_timers), save, public cputime

  accumulated cpu time in each timer
- character(len=8), dimension(max_timers), save, public status

  timer status string

15.140.1 Detailed Description

Timing routines.

This module uses F90 cpu time routines to allowing setting of multiple CPU timers.

Authors

Matthias Cuntz - from timers.f (c) the Regents of the University of Californi

Date

Dec 2012

Definition at line 11 of file mo_timer.f90.

15.140.2 Member Function/Subroutine Documentation

15.140.2.1 subroutine, public mo_timer::timer_check (integer(i4), intent(in) *timer*)

Check a timer.

This routine checks a given timer. This is primarily used to periodically accumulate time in the timer to prevent timer cycles from wrapping around max_cycles.

Parameters

in	*integer(i4) :: timer*	timer number

Author

Matthias Cuntz

Date

Aug 2012

Definition at line 156 of file mo_timer.f90.

References timer_start(), and timer_stop().

Here is the call graph for this function:

15.140.2.2 subroutine, public mo_timer::timer_clear (integer(i4), intent(in), optional *timer*)

Reset a timer.

This routine resets a given timer or all timers to 0.

Parameters

in	*integer(i4), optional :: timer*	timer number if given. If missing, all timers will be reset.

Author

Matthias Cuntz

Date

Aug 2012

Definition at line 212 of file mo_timer.f90.

15.140.2.3 real(sp) function, public mo_timer::timer_get (integer(i4), intent(in) *timer*)

Return a timer.

This routine returns the result of a given timer. This can be called instead of timer_print so that the calling routine can print it in desired format.

Parameters

in	*integer(i4) :: timer*	timer number

Author

Matthias Cuntz

Date

Aug 2012

Definition at line 269 of file mo_timer.f90.

References timer_start(), and timer_stop().

Referenced by mhm_driver(), mrm_driver(), mo_meteo_forcings::prepare_meteo_forcings_data(), mo_read_optional_data::read_evapotranspiration(), mo_read_optional_data::read_neutrons(), and mo_read_optional_data::read_soil_moisture().

Here is the call graph for this function:

Here is the caller graph for this function:

15.140.2.4 subroutine, public mo_timer::timer_print (integer(i4), intent(in) *timer*)

Print a timer.

This routine prints the accumulated cpu time in given timer.

Parameters

in	*integer(i4) :: timer*	timer number

Author

Matthias Cuntz

Date

Aug 2012

Definition at line 328 of file mo_timer.f90.

References timer_start(), and timer_stop().

Here is the call graph for this function:

15.140.2.5 subroutine, public mo_timer::timer_start (integer(i4), intent(in) *timer*)

Start a timer.

This routine starts a given timer.

Parameters

in	*integer(i4) :: timer*	timer number

Author

Matthias Cuntz

Date

Aug 2012

Definition at line 392 of file mo_timer.f90.

Referenced by mhm_driver(), mrm_driver(), mo_meteo_forcings::prepare_meteo_forcings_data(), mo_read↩_optional_data::read_evapotranspiration(), mo_read_optional_data::read_neutrons(), mo_read_optional_data↩::read_soil_moisture(), timer_check(), timer_get(), and timer_print().

Here is the caller graph for this function:

15.140.2.6 subroutine, public mo_timer::timer_stop (integer(i4), intent(in) *timer*)

Stop a timer.

This routine stops a given timer.

Parameters

in	*integer(i4) :: timer*	timer number

Author

Matthias Cuntz

Date

Aug 2012

Definition at line 450 of file mo_timer.f90.

Referenced by mhm_driver(), mrm_driver(), mo_meteo_forcings::prepare_meteo_forcings_data(), mo_read↩_optional_data::read_evapotranspiration(), mo_read_optional_data::read_neutrons(), mo_read_optional_data↩::read_soil_moisture(), timer_check(), timer_get(), and timer_print().

Here is the caller graph for this function:

15.140.2.7 subroutine, public mo_timer::timers_init ( )

Initialise timer module.

This routine initializes some machine parameters necessary for computing cpu time from F90 intrinsics.

Author

Matthias Cuntz

Date

Aug 2012

Definition at line 533 of file mo_timer.f90.

Referenced by mhm_driver(), and mrm_driver().

Here is the caller graph for this function:

15.140.3 Member Data Documentation

15.140.3.1 real(sp), save, public mo_timer::clock_rate

clock_rate in seconds for each cycle

Definition at line 99 of file mo_timer.f90.

15.140.3.2 real(sp), dimension(max_timers), save, public mo_timer::cputime

accumulated cpu time in each timer

Definition at line 105 of file mo_timer.f90.

15.140.3.3 integer(i4), dimension(max_timers), save, public mo_timer::cycles1

cycle number at start for each timer

Definition at line 101 of file mo_timer.f90.

15.140.3.4 integer(i4), dimension(max_timers), save, public mo_timer::cycles2

cycle number at stop for each timer

Definition at line 103 of file mo_timer.f90.

15.140.3.5 integer(i4), save, public mo_timer::cycles_max

max value of clock allowed by system

Definition at line 97 of file mo_timer.f90.

15.140.3.6 integer(i4), parameter, public mo_timer::max_timers = 99

max number of timers allowed

Definition at line 95 of file mo_timer.f90.

15.140.3.7 character(len = 8), dimension(max_timers), save, public mo_timer::status

timer status string

Definition at line 107 of file mo_timer.f90.

The documentation for this module was generated from the following file:

- mo_timer.f90

15.141 mo_utils Module Reference

General utilities for the CHS library.

Data Types

- interface eq
- interface equal

  Comparison of real values.
- interface ge
- interface greaterequal
- interface is_finite

  .true. if not IEEE Inf, IEEE NaN, nor IEEE Inf nor IEEE NaN, respectively.
- interface is_nan
- interface is_normal
- interface le
- interface lesserequal
- interface locate

  Find closest values in a monotonic series, returns the indexes.
- interface ne
- interface notequal
- interface special_value

  Special IEEE values.
- interface swap

  Swap to values or two elements in array.

Private Member Functions

- elemental pure logical function equal_dp (a, b)
- elemental pure logical function equal_sp (a, b)
- elemental pure logical function greaterequal_dp (a, b)
- elemental pure logical function greaterequal_sp (a, b)
- elemental pure logical function lesserequal_dp (a, b)
- elemental pure logical function lesserequal_sp (a, b)
- elemental pure logical function notequal_dp (a, b)
- elemental pure logical function notequal_sp (a, b)
- elemental pure logical function is_finite_dp (a)

- elemental pure logical function is_finite_sp (a)
- elemental pure logical function is_nan_dp (a)
- elemental pure logical function is_nan_sp (a)
- elemental pure logical function is_normal_dp (a)
- elemental pure logical function is_normal_sp (a)
- integer(i4) function locate_0d_dp (x, y)
- integer(i4) function locate_0d_sp (x, y)
- integer(i4) function,
  dimension(:), allocatable locate_1d_dp (x, y)
- integer(i4) function,
  dimension(:), allocatable locate_1d_sp (x, y)
- elemental pure subroutine swap_xy_dp (x, y)
- elemental pure subroutine swap_xy_sp (x, y)
- elemental pure subroutine swap_xy_i4 (x, y)
- subroutine swap_vec_dp (x, i1, i2)
- subroutine swap_vec_sp (x, i1, i2)
- subroutine swap_vec_i4 (x, i1, i2)
- real(dp) function special_value_dp (x, ieee)
- real(sp) function special_value_sp (x, ieee)

15.141.1 Detailed Description

General utilities for the CHS library.

This module provides general utilities such as comparisons of two reals.

Authors

Matthias Cuntz, Juliane Mai

Date

Feb 2014

Definition at line 9 of file mo_utils.f90.

15.141.2 Member Function/Subroutine Documentation

15.141.2.1 elemental pure logical function mo_utils::equal_dp (real(dp), intent(in) *a*, real(dp), intent(in) *b*) [private]

Definition at line 419 of file mo_utils.f90.

References mo_utils::equal::equal_dp().

Here is the call graph for this function:

15.141.2.2 elemental pure logical function mo_utils::equal_sp (real(sp), intent(in) *a*, real(sp), intent(in) *b*) [private]

Definition at line 436 of file mo_utils.f90.

References mo_utils::equal::equal_sp().

Here is the call graph for this function:

15.141.2.3 elemental pure logical function mo_utils::greaterequal_dp (real(dp), intent(in) *a*, real(dp), intent(in) *b*) [private]

Definition at line 454 of file mo_utils.f90.

References mo_utils::greaterequal::greaterequal_dp().

Here is the call graph for this function:

15.141.2.4 elemental pure logical function mo_utils::greaterequal_sp (real(sp), intent(in) *a*, real(sp), intent(in) *b*) [private]

Definition at line 469 of file mo_utils.f90.

References mo_utils::greaterequal::greaterequal_sp().

Here is the call graph for this function:

15.141.2.5 elemental pure logical function mo_utils::is_finite_dp (real(dp), intent(in) *a*) [private]

Definition at line 551 of file mo_utils.f90.

References mo_utils::is_finite::is_finite_dp().

Here is the call graph for this function:

15.141.2.6 elemental pure logical function mo_utils::is_finite_sp (real(sp), intent(in) *a*) [private]

Definition at line 570 of file mo_utils.f90.

References mo_utils::is_finite::is_finite_sp().

Here is the call graph for this function:

15.141.2.7 elemental pure logical function mo_utils::is_nan_dp (real(dp), intent(in) *a*) [private]

Definition at line 590 of file mo_utils.f90.

References mo_utils::is_nan::is_nan_dp().

Here is the call graph for this function:

15.141.2.8 elemental pure logical function mo_utils::is_nan_sp (real(sp), intent(in) *a*) [private]

Definition at line 610 of file mo_utils.f90.

References mo_utils::is_nan::is_nan_sp().

Here is the call graph for this function:

15.141.2.9 elemental pure logical function mo_utils::is_normal_dp (real(dp), intent(in) *a*) [private]

Definition at line 631 of file mo_utils.f90.

References mo_utils::is_normal::is_normal_dp().

Here is the call graph for this function:

15.141.2.10 elemental pure logical function mo_utils::is_normal_sp (real(sp), intent(in) *a*) [private]

Definition at line 650 of file mo_utils.f90.

References mo_utils::is_normal::is_normal_sp().

Here is the call graph for this function:

15.141.2.11 elemental pure logical function mo_utils::lesserequal_dp (real(dp), intent(in) *a*, real(dp), intent(in) *b*) [private]

Definition at line 485 of file mo_utils.f90.

References mo_utils::lesserequal::lesserequal_dp().

Here is the call graph for this function:

15.141.2.12 elemental pure logical function mo_utils::lesserequal_sp (real(sp), intent(in) *a*, real(sp), intent(in) *b*) [private]

Definition at line 500 of file mo_utils.f90.

References mo_utils::lesserequal::lesserequal_sp().

Here is the call graph for this function:

15.141.2.13 integer(i4) function mo_utils::locate_0d_dp (real(dp), dimension(:), intent(in) *x*, real(dp), intent(in) *y*) [private]

Definition at line 675 of file mo_utils.f90.

References mo_utils::locate::locate_0d_dp().

Here is the call graph for this function:

15.141.2.14 integer(i4) function mo_utils::locate_0d_sp (real(sp), dimension(:), intent(in) *x*, real(sp), intent(in) *y*) [private]

Definition at line 694 of file mo_utils.f90.

References mo_utils::locate::locate_0d_sp().

Here is the call graph for this function:

15.141.2.15 integer(i4) function, dimension(:), allocatable mo_utils::locate_1d_dp (real(dp), dimension(:), intent(in) *x*, real(dp), dimension(:), intent(in) *y*) [private]

Definition at line 713 of file mo_utils.f90.

References mo_utils::locate::locate_1d_dp().

Here is the call graph for this function:

15.141.2.16 integer(i4) function, dimension(:), allocatable mo_utils::locate_1d_sp (real(sp), dimension(:), intent(in) *x*, real(sp), dimension(:), intent(in) *y*) [private]

Definition at line 738 of file mo_utils.f90.

References mo_utils::locate::locate_1d_sp().

Here is the call graph for this function:

15.141.2.17 elemental pure logical function mo_utils::notequal_dp (real(dp), intent(in) *a*, real(dp), intent(in) *b*) [private]

Definition at line 516 of file mo_utils.f90.

References mo_utils::notequal::notequal_dp().

Here is the call graph for this function:

15.141.2.18 elemental pure logical function mo_utils::notequal_sp (real(sp), intent(in) *a*, real(sp), intent(in) *b*) [private]

Definition at line 533 of file mo_utils.f90.

References mo_utils::notequal::notequal_sp().

Here is the call graph for this function:

15.141.2.19 real(dp) function mo_utils::special_value_dp (real(dp), intent(in) *x*, character(len = ∗), intent(in) *ieee*) [private]

Definition at line 861 of file mo_utils.f90.

References mo_utils::special_value::special_value_dp(), and mo_string_utils::toupper().

Here is the call graph for this function:

15.141.2.20 real(sp) function mo_utils::special_value_sp (real(sp), intent(in) *x*, character(len = ∗), intent(in) *ieee*) [private]

Definition at line 948 of file mo_utils.f90.

References mo_utils::special_value::special_value_sp(), and mo_string_utils::toupper().

Here is the call graph for this function:

15.141.2.21 subroutine mo_utils::swap_vec_dp (real(dp), dimension(:), intent(inout) *x*, integer(i4), intent(in) *i1*, integer(i4), intent(in) *i2*) [private]

Definition at line 811 of file mo_utils.f90.

15.141.2.22 subroutine mo_utils::swap_vec_i4 (integer(i4), dimension(:), intent(inout) *x*, integer(i4), intent(in) *i1*, integer(i4), intent(in) *i2*) [private]

Definition at line 843 of file mo_utils.f90.

15.141.2.23 subroutine mo_utils::swap_vec_sp (real(sp), dimension(:), intent(inout) *x*, integer(i4), intent(in) *i1*, integer(i4), intent(in) *i2*) [private]

Definition at line 827 of file mo_utils.f90.

15.141.2.24 elemental pure subroutine mo_utils::swap_xy_dp (real(dp), intent(inout) *x*, real(dp), intent(inout) *y*) [private]

Definition at line 765 of file mo_utils.f90.

15.141.2.25 elemental pure subroutine mo_utils::swap_xy_i4 (integer(i4), intent(inout) *x*, integer(i4), intent(inout) *y*) [private]

Definition at line 795 of file mo_utils.f90.

15.141.2.26 elemental pure subroutine mo_utils::swap_xy_sp (real(sp), intent(inout) *x*, real(sp), intent(inout) *y*) [private]

Definition at line 780 of file mo_utils.f90.

The documentation for this module was generated from the following file:

- mo_utils.f90

15.142 mo_write_ascii Module Reference

Module to write ascii file output.

Public Member Functions

- 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.

15.142.1 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

Definition at line 19 of file mo_write_ascii.f90.

15.142.2 Member Function/Subroutine Documentation

15.142.2.1 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_message::message().

Referenced by mhm_driver(), and mo_mrm_write::mrm_write().

Here is the call graph for this function:

Here is the caller graph for this function:

15.142.2.2 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 433 of file mo_write_ascii.f90.

References mo_message::message().

Referenced by mhm_driver().

Here is the call graph for this function:

Here is the caller graph for this function:

15.142.2.3 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(n↩ Processes, 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 517 of file mo_write_ascii.f90.

References mo_message::message().

Referenced by mhm_driver().

Here is the call graph for this function:

Here is the caller graph for this function:

The documentation for this module was generated from the following file:

- mo_write_ascii.f90

15.143 mo_write_fluxes_states Module Reference

Creates NetCDF output for different fluxes and state variables of mHM.

Data Types

- interface outputdataset
- interface outputvariable

Public Member Functions

- type(outputdataset) function,
  public newoutputdataset (ibasin, 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.
- type(ncdataset) function createoutputfile (ibasin)

  Create and initialize output file.
- subroutine writevariableattributes (var, long_name, unit)

  Write output variable attributes.

Private Member Functions

- 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.
- subroutine close (self)

  Close the file.
- character(16) function fluxesunit (ibasin)

  Generate a unit string.

15.143.1 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

Definition at line 15 of file mo_write_fluxes_states.f90.

15.143.2 Member Function/Subroutine Documentation

15.143.2.1 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 838 of file mo_write_fluxes_states.f90.

References mo_message::message().

Here is the call graph for this function:

15.143.2.2 type(ncdataset) function mo_write_fluxes_states::createoutputfile (integer(i4), intent(in) *ibasin*)

Create and initialize output file.

Create output file, write all non-dynamic variables and global attributes for the given basin.

Returns

type(NcDataset)

Parameters

in	*integer(i4) :: ibasin*	-> basin id

Authors

David Schaefer

Date

June 2015

Definition at line 879 of file mo_write_fluxes_states.f90.

References mo_julian::dec2date(), mo_grid::geocoordinates(), and mo_grid::mapcoordinates().

Referenced by mo_mrm_write_fluxes_states::newoutputdataset(), and newoutputdataset().

Here is the call graph for this function:

Here is the caller graph for this function:

15.143.2.3 character(16) function mo_write_fluxes_states::fluxesunit (integer(i4), intent(in) *ibasin*) [private]

Generate a unit string.

Generate the unit string for the output variable netcdf attribute based on modeling timestep

Returns

character(16)

Parameters

in	*integer(i4) :: ibasin*	

Authors

David Schaefer

Date

June 2015

Definition at line 1035 of file mo_write_fluxes_states.f90.

Referenced by newoutputdataset().

Here is the caller graph for this function:

15.143.2.4 type(outputdataset) function, public mo_write_fluxes_states::newoutputdataset (integer(i4), intent(in) *ibasin,* logical, dimension(:, :), intent(in), pointer *mask1,* integer(i4), intent(in) *nCells*)

Initialize OutputDataset.

Create and initialize the output file. If new a new output variable needs to be written, this is the first of two procedures to change (second: updateDataset)

Returns

type(OutputDataset)

Parameters

in	*integer(i4) :: ibasin*	-> basin id
in	*logical, dimension(:, :) :: mask1*	-> L1 mask to reconstruct the data
in	*integer(i4) :: nCells*	

Authors

Matthias Zink

Date

Apr 2013

Definition at line 240 of file mo_write_fluxes_states.f90.

References createoutputfile(), fluxesunit(), and writevariableattributes().

Referenced by mo_mrm_write_fluxes_states::outputdataset::close(), 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:

15.143.2.5 type(outputvariable) function mo_write_fluxes_states::newoutputvariable (type(ncdataset), intent(in) *nc,* character(∗), intent(in) *name,* character(∗), intent(in) *dtype,* character(16), dimension(3), intent(in) *dims,* integer(i4), intent(in) *ncells,* logical, dimension(:, :), intent(in), target *mask,* logical, intent(in), optional *avg*) [private]

Initialize OutputVariable.

TODO: add description

Returns

type(OutputVariable)

Parameters

in	*type(NcDataset) :: nc*	-> NcDataset which contains the variable
in	*character(∗) :: name*	
in	*character(∗) :: dtype*	
in	*character(16), dimension(3) :: dims*	
in	*integer(i4) :: ncells*	-> number of cells in basin
in	*logical, dimension(:, :) :: mask*	
in	*logical, optional :: avg*	-> average the data before writing

Authors

David Schaefer

Date

June 2015

Definition at line 102 of file mo_write_fluxes_states.f90.

Referenced by mo_mrm_write_fluxes_states::outputvariable::writevariabletimestep(), and mo_write_fluxes_states::outputvariable::writevariabletimestep().

Here is the caller graph for this function:

15.143.2.6 subroutine mo_write_fluxes_states::updatedataset (class(outputdataset), intent(inout), target *self,* integer(i4), intent(in) *sidx,* integer(i4), intent(in) *eidx,* real(dp), dimension(:), intent(in) *L1_fSealed,* real(dp), dimension(:), intent(in) *L1_fNotSealed,* real(dp), dimension(:), intent(in) *L1_inter,* real(dp), dimension(:), intent(in) *L1_snowPack,* real(dp), dimension(:, :), intent(in) *L1_soilMoist,* real(dp), dimension(:, :), intent(in) *L1_soilMoistSat,* real(dp), dimension(:), intent(in) *L1_sealSTW,* real(dp), dimension(:), intent(in) *L1_unsatSTW,* real(dp), dimension(:), intent(in) *L1_satSTW,* real(dp), dimension(:), intent(in) *L1_neutrons,* real(dp), dimension(:), intent(in) *L1_pet,* real(dp), dimension(:, :), intent(in) *L1_aETSoil,* real(dp), dimension(:), intent(in) *L1_aETCanopy,* real(dp), dimension(:), intent(in) *L1_aETSealed,* real(dp), dimension(:), intent(in) *L1_total_runoff,* real(dp), dimension(:), intent(in) *L1_runoffSeal,* real(dp), dimension(:), intent(in) *L1_fastRunoff,* real(dp), dimension(:), intent(in) *L1_slowRunoff,* real(dp), dimension(:), intent(in) *L1_baseflow,* real(dp), dimension(:), intent(in) *L1_percol,* real(dp), dimension(:, :), intent(in) *L1_infilSoil,* real(dp), dimension(:), intent(in) *L1_preEffect*)

Update all variables.

Call the type bound procedure updateVariable for all output variables. If a new output variable needs to be written, this is the second of two procedures to change (first: newOutputDataset)

Parameters

in,out	*class(OutputDataset) :: self*	
in	*integer(i4) :: sidx, eidx*	
in	*integer(i4) :: sidx, eidx*	
in	*real(dp), dimension(:) :: L1_fSealed*	
in	*real(dp), dimension(:) :: L1_fNotSealed*	
in	*real(dp), dimension(:) :: L1_inter*	
in	*real(dp), dimension(:) :: L1_snowPack*	
in	*real(dp), dimension(:, :) :: L1_soilMoist*	
in	*real(dp), dimension(:, :) :: L1_soilMoistSat*	
in	*real(dp), dimension(:) :: L1_sealSTW*	
in	*real(dp), dimension(:) :: L1_unsatSTW*	
in	*real(dp), dimension(:) :: L1_satSTW*	
in	*real(dp), dimension(:) :: L1_neutrons*	
in	*real(dp), dimension(:) :: L1_pet*	
in	*real(dp), dimension(:, :) :: L1_aETSoil*	
in	*real(dp), dimension(:) :: L1_aETCanopy*	

in	*real(dp), dimension(:) :: L1_aETSealed*	
in	*real(dp), dimension(:) :: L1_total_runoff*	
in	*real(dp), dimension(:) :: L1_runoffSeal*	
in	*real(dp), dimension(:) :: L1_fastRunoff*	
in	*real(dp), dimension(:) :: L1_slowRunoff*	
in	*real(dp), dimension(:) :: L1_baseflow*	
in	*real(dp), dimension(:) :: L1_percol*	
in	*real(dp), dimension(:, :) :: L1_infilSoil*	
in	*real(dp), dimension(:) :: L1_preEffect*	

Authors

Matthias Zink

Date

Apr 2013

Definition at line 507 of file mo_write_fluxes_states.f90.

References mo_runoff::l1_total_runoff(), and mo_write_fluxes_states::outputvariable::updatevariable().

Here is the call graph for this function:

15.143.2.7 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 158 of file mo_write_fluxes_states.f90.

15.143.2.8 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 795 of file mo_write_fluxes_states.f90.

15.143.2.9 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 994 of file mo_write_fluxes_states.f90.

Referenced by mo_mrm_write_fluxes_states::newoutputdataset(), and newoutputdataset().

Here is the caller graph for this function:

15.143.2.10 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 195 of file mo_write_fluxes_states.f90.

The documentation for this module was generated from the following file:

- mo_write_fluxes_states.f90

15.144 mo_xor4096 Module Reference

Collaboration diagram for mo_xor4096:

Data Types

- interface get_timeseed
- interface xor4096
- interface xor4096g

Public Attributes

- integer(i4), parameter, public n_save_state = 132_i4

  Dimension of vector saving the state of a stream.

Private Member Functions

- subroutine get_timeseed_i4_0d (seed)
- subroutine get_timeseed_i4_1d (seed)
- subroutine get_timeseed_i8_0d (seed)
- subroutine get_timeseed_i8_1d (seed)
- subroutine xor4096s_0d (seed, SingleIntegerRN, save_state)
- subroutine xor4096s_1d (seed, SingleIntegerRN, save_state)
- subroutine xor4096f_0d (seed, SingleRealRN, save_state)

- subroutine xor4096f_1d (seed, SingleRealRN, save_state)
- subroutine xor4096l_0d (seed, DoubleIntegerRN, save_state)
- subroutine xor4096l_1d (seed, DoubleIntegerRN, save_state)
- subroutine xor4096d_0d (seed, DoubleRealRN, save_state)
- subroutine xor4096d_1d (seed, DoubleRealRN, save_state)
- subroutine xor4096gf_0d (seed, SingleRealRN, save_state)
- subroutine xor4096gf_1d (seed, SingleRealRN, save_state)
- subroutine xor4096gd_0d (seed, DoubleRealRN, save_state)
- subroutine xor4096gd_1d (seed, DoubleRealRN, save_state)

15.144.1 Detailed Description

Definition at line 1 of file mo_xor4096.f90.

15.144.2 Member Function/Subroutine Documentation

15.144.2.1 subroutine mo_xor4096::get_timeseed_i4_0d (integer(i4), intent(inout) *seed*) [private]

Definition at line 363 of file mo_xor4096.f90.

15.144.2.2 subroutine mo_xor4096::get_timeseed_i4_1d (integer(i4), dimension(:), intent(inout) *seed*) [private]

Definition at line 380 of file mo_xor4096.f90.

15.144.2.3 subroutine mo_xor4096::get_timeseed_i8_0d (integer(i8), intent(inout) *seed*) [private]

Definition at line 401 of file mo_xor4096.f90.

15.144.2.4 subroutine mo_xor4096::get_timeseed_i8_1d (integer(i8), dimension(:), intent(inout) *seed*) [private]

Definition at line 418 of file mo_xor4096.f90.

15.144.2.5 subroutine mo_xor4096::xor4096d_0d (integer(i8), intent(in) *seed,* real(dp), intent(out) *DoubleRealRN,* integer(i8), dimension(**n_save_state**), intent(inout), optional *save_state*) [private]

Definition at line 1134 of file mo_xor4096.f90.

15.144.2.6 subroutine mo_xor4096::xor4096d_1d (integer(i8), dimension(:), intent(in) *seed,* real(dp), dimension(size(seed, 1)), intent(out) *DoubleRealRN,* integer(i8), dimension(size(seed, 1), **n_save_state**), intent(inout), optional *save_state*) [private]

Definition at line 1237 of file mo_xor4096.f90.

15.144.2.7 subroutine mo_xor4096::xor4096f_0d (integer(i4), intent(in) *seed,* real(sp), intent(out) *SingleRealRN,* integer(i4), dimension(**n_save_state**), intent(inout), optional *save_state*) [private]

Definition at line 665 of file mo_xor4096.f90.

15.144.2.8 subroutine mo_xor4096::xor4096f_1d (integer(i4), dimension(:), intent(in) *seed,* real(sp), dimension(size(seed)), intent(out) *SingleRealRN,* integer(i4), dimension(size(seed, 1), **n_save_state**), intent(inout), optional *save_state*) [private]

Definition at line 769 of file mo_xor4096.f90.

15.144.2.9 subroutine mo_xor4096::xor4096gd_0d (integer(i8), intent(in) *seed,* real(dp), intent(out) *DoubleRealRN,* integer(i8), dimension(**n_save_state**), intent(inout), optional *save_state*) [private]

Definition at line 1742 of file mo_xor4096.f90.

15.144.2.10 subroutine mo_xor4096::xor4096gd_1d (integer(i8), dimension(:), intent(in) *seed,* real(dp), dimension(size(seed)), intent(out) *DoubleRealRN,* integer(i8), dimension(size(seed), **n_save_state**), intent(inout), optional *save_state*) [private]

Definition at line 1897 of file mo_xor4096.f90.

15.144.2.11 subroutine mo_xor4096::xor4096gf_0d (integer(i4), intent(in) *seed,* real(sp), intent(out) *SingleRealRN,* integer(i4), dimension(**n_save_state**), intent(inout), optional *save_state*) [private]

Definition at line 1377 of file mo_xor4096.f90.

15.144.2.12 subroutine mo_xor4096::xor4096gf_1d (integer(i4), dimension(:), intent(in) *seed,* real(sp), dimension(size(seed)), intent(out) *SingleRealRN,* integer(i4), dimension(size(seed), **n_save_state**), intent(inout), optional *save_state*) [private]

Definition at line 1533 of file mo_xor4096.f90.

15.144.2.13 subroutine mo_xor4096::xor4096l_0d (integer(i8), intent(in) *seed,* integer(i8), intent(out) *DoubleIntegerRN,* integer(i8), dimension(**n_save_state**), intent(inout), optional *save_state*) [private]

Definition at line 910 of file mo_xor4096.f90.

15.144.2.14 subroutine mo_xor4096::xor4096l_1d (integer(i8), dimension(:), intent(in) *seed,* integer(i8), dimension(size(seed, 1)), intent(out) *DoubleIntegerRN,* integer(i8), dimension(size(seed, 1), **n_save_state**), intent(inout), optional *save_state*) [private]

Definition at line 1003 of file mo_xor4096.f90.

15.144.2.15 subroutine mo_xor4096::xor4096s_0d (integer(i4), intent(in) *seed,* integer(i4), intent(out) *SingleIntegerRN,* integer(i4), dimension(**n_save_state**), intent(inout), optional *save_state*) [private]

Definition at line 441 of file mo_xor4096.f90.

15.144.2.16 subroutine mo_xor4096::xor4096s_1d (integer(i4), dimension(:), intent(in) *seed,* integer(i4), dimension(size(seed, 1)), intent(out) *SingleIntegerRN,* integer(i4), dimension(size(seed, 1), **n_save_state**), intent(inout), optional *save_state*) [private]

Definition at line 534 of file mo_xor4096.f90.

15.144.3 Member Data Documentation

15.144.3.1 integer(i4), parameter, public mo_xor4096::n_save_state = 132_i4

Dimension of vector saving the state of a stream.

Definition at line 355 of file mo_xor4096.f90.

The documentation for this module was generated from the following file:

- mo_xor4096.f90

15.145 mo_moment::moment Interface Reference

Public Member Functions

- subroutine moment_sp (dat, average, variance, skewness, kurtosis, mean, stddev, absdev, mask)
- subroutine moment_dp (dat, average, variance, skewness, kurtosis, mean, stddev, absdev, mask)

15.145.1 Detailed Description

Definition at line 672 of file mo_moment.f90.

15.145.2 Member Function/Subroutine Documentation

15.145.2.1 subroutine mo_moment::moment::moment_dp (real(dp), dimension(:), intent(in) *dat,* real(dp), intent(out), optional *average,* real(dp), intent(out), optional *variance,* real(dp), intent(out), optional *skewness,* real(dp), intent(out), optional *kurtosis,* real(dp), intent(out), optional *mean,* real(dp), intent(out), optional *stddev,* real(dp), intent(out), optional *absdev,* logical, dimension(:), intent(in), optional *mask*)

Definition at line 1589 of file mo_moment.f90.

15.145.2.2 subroutine mo_moment::moment::moment_sp (real(sp), dimension(:), intent(in) *dat,* real(sp), intent(out), optional *average,* real(sp), intent(out), optional *variance,* real(sp), intent(out), optional *skewness,* real(sp), intent(out), optional *kurtosis,* real(sp), intent(out), optional *mean,* real(sp), intent(out), optional *stddev,* real(sp), intent(out), optional *absdev,* logical, dimension(:), intent(in), optional *mask*)

Definition at line 1659 of file mo_moment.f90.

The documentation for this interface was generated from the following file:

- mo_moment.f90

15.146 mo_orderpack::mrgref Interface Reference

Public Member Functions

- subroutine d_mrgref (XVALT, IRNGT)
- subroutine r_mrgref (XVALT, IRNGT)
- subroutine i_mrgref (XVALT, IRNGT)

15.146.1 Detailed Description

Definition at line 312 of file mo_orderpack.f90.

15.146.2 Member Function/Subroutine Documentation

15.146.2.1 subroutine mo_orderpack::mrgref::d_mrgref (real(kind = dp), dimension (:), intent(in) *XVALT,* integer(kind = i4), dimension (:), intent(out) *IRNGT*)

Definition at line 5228 of file mo_orderpack.f90.

15.146.2.2 subroutine mo_orderpack::mrgref::i_mrgref (integer(kind = i4), dimension (:), intent(in) *XVALT,* integer(kind = i4), dimension (:), intent(out) *IRNGT*)

Definition at line 5486 of file mo_orderpack.f90.

15.146.2.3 subroutine mo_orderpack::mrgref::r_mrgref (real(kind = sp), dimension (:), intent(in) *XVALT,* integer(kind = i4), dimension (:), intent(out) *IRNGT*)

Definition at line 5357 of file mo_orderpack.f90.

The documentation for this interface was generated from the following file:

- mo_orderpack.f90

15.147 mo_orderpack::mrgrnk Interface Reference

Public Member Functions

- subroutine d_mrgrnk (XDONT, IRNGT)
- subroutine r_mrgrnk (XDONT, IRNGT)
- subroutine i_mrgrnk (XDONT, IRNGT)

15.147.1 Detailed Description

Definition at line 315 of file mo_orderpack.f90.

15.147.2 Member Function/Subroutine Documentation

15.147.2.1 subroutine mo_orderpack::mrgrnk::d_mrgrnk (real(kind = dp), dimension (:), intent(in) *XDONT,* integer(kind = i4), dimension (:), intent(out) *IRNGT*)

Definition at line 5615 of file mo_orderpack.f90.

15.147.2.2 subroutine mo_orderpack::mrgrnk::i_mrgrnk (integer(kind = i4), dimension (:), intent(in) *XDONT,* integer(kind = i4), dimension (:), intent(out) *IRNGT*)

Definition at line 6015 of file mo_orderpack.f90.

15.147.2.3 subroutine mo_orderpack::mrgrnk::r_mrgrnk (real(kind = sp), dimension (:), intent(in) *XDONT*, integer(kind = i4), dimension (:), intent(out) *IRNGT*)

Definition at line 5815 of file mo_orderpack.f90.

The documentation for this interface was generated from the following file:

- mo_orderpack.f90

15.148 mo_errormeasures::mse Interface Reference

Public Member Functions

- real(sp) function mse_sp_1d (x, y, mask)
- real(dp) function mse_dp_1d (x, y, mask)
- real(sp) function mse_sp_2d (x, y, mask)
- real(dp) function mse_dp_2d (x, y, mask)
- real(sp) function mse_sp_3d (x, y, mask)
- real(dp) function mse_dp_3d (x, y, mask)

15.148.1 Detailed Description

Definition at line 456 of file mo_errormeasures.f90.

15.148.2 Member Function/Subroutine Documentation

15.148.2.1 real(dp) function mo_errormeasures::mse::mse_dp_1d (real(dp), dimension(:), intent(in) *x*, real(dp), dimension(:), intent(in) *y*, logical, dimension(:), intent(in), optional *mask*)

Definition at line 2292 of file mo_errormeasures.f90.

Referenced by mo_errormeasures::mse_dp_1d(), and mo_errormeasures::rmse_dp_1d().

Here is the caller graph for this function:

15.148.2.2 real(dp) function mo_errormeasures::mse::mse_dp_2d (real(dp), dimension(:, :), intent(in) *x*, real(dp), dimension(:, :), intent(in) *y*, logical, dimension(:, :), intent(in), optional *mask*)

Definition at line 2358 of file mo_errormeasures.f90.

Referenced by mo_errormeasures::mse_dp_2d(), and mo_errormeasures::rmse_dp_2d().

Here is the caller graph for this function:

15.148.2.3 real(dp) function mo_errormeasures::mse::mse_dp_3d (real(dp), dimension(:, :, :), intent(in) *x*, real(dp), dimension(:, :, :), intent(in) *y*, logical, dimension(:, :, :), intent(in), optional *mask*)

Definition at line 2425 of file mo_errormeasures.f90.

Referenced by mo_errormeasures::mse_dp_3d(), and mo_errormeasures::rmse_dp_3d().

Here is the caller graph for this function:

15.148.2.4 real(sp) function mo_errormeasures::mse::mse_sp_1d (real(sp), dimension(:), intent(in) *x*, real(sp), dimension(:), intent(in) *y*, logical, dimension(:), intent(in), optional *mask*)

Definition at line 2259 of file mo_errormeasures.f90.

Referenced by mo_errormeasures::mse_sp_1d(), and mo_errormeasures::rmse_sp_1d().

Here is the caller graph for this function:

15.148.2.5 real(sp) function mo_errormeasures::mse::mse_sp_2d (real(sp), dimension(:, :), intent(in) *x*, real(sp), dimension(:, :), intent(in) *y*, logical, dimension(:, :), intent(in), optional *mask*)

Definition at line 2325 of file mo_errormeasures.f90.

Referenced by mo_errormeasures::mse_sp_2d(), and mo_errormeasures::rmse_sp_2d().

Here is the caller graph for this function:

15.148.2.6 real(sp) function mo_errormeasures::mse::mse_sp_3d (real(sp), dimension(:, :, :), intent(in) *x*, real(sp), dimension(:, :, :), intent(in) *y*, logical, dimension(:, :, :), intent(in), optional *mask*)

Definition at line 2391 of file mo_errormeasures.f90.

Referenced by mo_errormeasures::mse_sp_3d(), and mo_errormeasures::rmse_sp_3d().

Here is the caller graph for this function:

The documentation for this interface was generated from the following file:

- mo_errormeasures.f90

15.149 mo_orderpack::mulcnt Interface Reference

Public Member Functions

- subroutine d_mulcnt (XDONT, IMULT)
- subroutine r_mulcnt (XDONT, IMULT)
- subroutine i_mulcnt (XDONT, IMULT)

15.149.1 Detailed Description

Definition at line 318 of file mo_orderpack.f90.

15.149.2 Member Function/Subroutine Documentation

15.149.2.1 subroutine mo_orderpack::mulcnt::d_mulcnt (real(kind = dp), dimension (:), intent(in) *XDONT,* integer(kind = i4), dimension (:), intent(out) *IMULT*)

Definition at line 6215 of file mo_orderpack.f90.

15.149.2.2 subroutine mo_orderpack::mulcnt::i_mulcnt (integer(kind = i4), dimension (:), intent(in) *XDONT,* integer(kind = i4), dimension (:), intent(out) *IMULT*)

Definition at line 6269 of file mo_orderpack.f90.

15.149.2.3 subroutine mo_orderpack::mulcnt::r_mulcnt (real(kind = sp), dimension (:), intent(in) *XDONT,* integer(kind = i4), dimension (:), intent(out) *IMULT*)

Definition at line 6242 of file mo_orderpack.f90.

The documentation for this interface was generated from the following file:

- mo_orderpack.f90

15.150 mo_percentile::n_element Interface Reference

Public Member Functions

- real(dp) function n_element_dp (idat, n, mask, before, after, previous, next)
- real(sp) function n_element_sp (idat, n, mask, before, after, previous, next)

15.150.1 Detailed Description

Definition at line 143 of file mo_percentile.f90.

15.150.2 Member Function/Subroutine Documentation

15.150.2.1 real(dp) function mo_percentile::n_element::n_element_dp (real(dp), dimension(:), intent(in) *idat,* integer(i4), intent(in) *n,* logical, dimension(:), intent(in), optional *mask,* real(dp), intent(out), optional *before,* real(dp), intent(out), optional *after,* real(dp), intent(out), optional *previous,* real(dp), intent(out), optional *next*)

Definition at line 366 of file mo_percentile.f90.

Referenced by mo_percentile::n_element_dp().

Here is the caller graph for this function:

15.150.2.2 real(sp) function mo_percentile::n_element::n_element_sp (real(sp), dimension(:), intent(in) *idat,* integer(i4), intent(in) *n,* logical, dimension(:), intent(in), optional *mask,* real(sp), intent(out), optional *before,* real(sp), intent(out), optional *after,* real(sp), intent(out), optional *previous,* real(sp), intent(out), optional *next*)

Definition at line 441 of file mo_percentile.f90.

Referenced by mo_percentile::n_element_sp().

Here is the caller graph for this function:

The documentation for this interface was generated from the following file:

- mo_percentile.f90

15.151 mo_netcdf::ncdataset Interface Reference

Provides basic file modification functionality.

Collaboration diagram for mo_netcdf::ncdataset:

Public Member Functions

- procedure, public initncdataset
- procedure, public getnovariables
- procedure, public getvariableids
- procedure, public getvariables
- procedure, public hasvariable

  Check if variable exists.
- procedure, public hasdimension

  Check if dimension exists.
- procedure, public getunlimiteddimension

  Return the unlimited dimension of the dataset.
- procedure, public isunlimited => isDatasetUnlimited

  Check if the dataset is unlimited.
- procedure, public close

  Close the datset.
- procedure, public setdimension

  Create a new dimension.
- generic, public setattribute =>setGlobalAttributeChar,setGlobalAttributeI8,setGlobalAttributeI16,setGlobal↩AttributeI32,setGlobalAttributeI64,setGlobalAttributeF32,setGlobalAttributeF64

  Create a new global Attribute.
- generic, public getattribute =>getGlobalAttributeChar,getGlobalAttributeI8,getGlobalAttributeI16,getGlobal↩AttributeI32,getGlobalAttributeI64,getGlobalAttributeF32,getGlobalAttributeF64

  Retrieve global attribute value.

- generic, public getdimension =>getDimensionById,getDimensionByName

  Retrieve NcDimension.
- generic, public setvariable =>setVariableWithNames,setVariableWithTypes,setVariableWithIds

  Create a NetCDF variable.
- generic, public getvariable =>getVariableByName

  Retrieve NcVariable.

Public Attributes

- character(256) fname
- character(256) filename
- character(256) of
- character(256) the
- character(256) opened
- character(256) dataset
- character(1) mode
- character(1) file
- character(1) open
- integer(i4) id
- integer(i4) netcdf

Private Member Functions

- procedure, private setglobalattributechar
- procedure, private setglobalattributei8
- procedure, private setglobalattributei16
- procedure, private setglobalattributei32
- procedure, private setglobalattributei64
- procedure, private setglobalattributef32
- procedure, private setglobalattributef64
- procedure, private getglobalattributechar
- procedure, private getglobalattributei8
- procedure, private getglobalattributei16
- procedure, private getglobalattributei32
- procedure, private getglobalattributei64
- procedure, private getglobalattributef32
- procedure, private getglobalattributef64
- procedure, private getdimensionbyname
- procedure, private getdimensionbyid
- procedure, private setvariablewithtypes
- procedure, private setvariablewithnames
- procedure, private setvariablewithids
- procedure, private getvariablebyname

15.151.1 Detailed Description

Provides basic file modification functionality.

Bound to this derived type is the basic file level create/retrieve functionality, i.e. functions/subroutines to create/retrieve dimensions, variables and global attributes. All files created by this derived type and its procedures are are NF90_NETCDF4 only. The supported modes are: r: read w: write/create

Parameters

in	*character(∗) :: fname*	
in	*character(1) :: mode*	

Returns

"type(NcDataset)"

Definition at line 94 of file mo_netcdf.f90.

15.151.2 Member Function/Subroutine Documentation

15.151.2.1 procedure, public mo_netcdf::ncdataset::close ( )

Close the datset.

Close the NetCDF datset. The program will teminate abruptly if the file cannot be closed correctly.

Author

David Schaefer

Date

June 2015

Definition at line 291 of file mo_netcdf.f90.

15.151.2.2 generic, public mo_netcdf::ncdataset::getattribute ( )

Retrieve global attribute value.

Retrieve the value for a global attribute specified by its name. The program will teminate abruptly if the attribute does not exist.

Parameters

in	*character(∗) :: name*	
out	*character(∗)/integer(i4)/real(sp)/real(dp) :: value*	

Author

David Schaefer

Date

June 2015

Definition at line 399 of file mo_netcdf.f90.

References getglobalattributechar(), getglobalattributef32(), getglobalattributef64(), getglobalattributei16(), getglobalattributei32(), getglobalattributei64(), and getglobalattributei8().

Referenced by mo_netcdf::ncdimension::getattribute().

Here is the call graph for this function:

Here is the caller graph for this function:

15.151.2.3 generic, public mo_netcdf::ncdataset::getdimension ( )

Retrieve NcDimension.

Retrieve the NcDimension derived type for the dimension specified by its name or id. The program will teminate abruptly if no such dimension exists.

Parameters

in	*character(∗)/integer(i4) :: name/id*	

Returns

NcDimension

Author

David Schaefer

Date

June 2015

Definition at line 439 of file mo_netcdf.f90.

References getdimensionbyid(), and getdimensionbyname().

Here is the call graph for this function:

15.151.2.4 procedure, private mo_netcdf::ncdataset::getdimensionbyid ( ) [private]

Definition at line 125 of file mo_netcdf.f90.

Referenced by getdimension(), and mo_netcdf::getdimensionbyid().

Here is the caller graph for this function:

15.151.2.5 procedure, private mo_netcdf::ncdataset::getdimensionbyname ( ) [private]

Definition at line 124 of file mo_netcdf.f90.

Referenced by getdimension(), and mo_netcdf::getdimensionbyname().

Here is the caller graph for this function:

15.151.2.6 procedure, private mo_netcdf::ncdataset::getglobalattributechar ( ) [private]

Definition at line 116 of file mo_netcdf.f90.

Referenced by getattribute().

Here is the caller graph for this function:

15.151.2.7 procedure, private mo_netcdf::ncdataset::getglobalattributef32 ( ) [private]

Definition at line 121 of file mo_netcdf.f90.

Referenced by getattribute().

Here is the caller graph for this function:

15.151.2.8 procedure, private mo_netcdf::ncdataset::getglobalattributef64 ( ) [private]

Definition at line 122 of file mo_netcdf.f90.

Referenced by getattribute().

Here is the caller graph for this function:

15.151.2.9 procedure, private mo_netcdf::ncdataset::getglobalattributei16 ( ) [private]

Definition at line 118 of file mo_netcdf.f90.

Referenced by getattribute().

Here is the caller graph for this function:

15.151.2.10 procedure, private mo_netcdf::ncdataset::getglobalattributei32 ( ) [private]

Definition at line 119 of file mo_netcdf.f90.

Referenced by getattribute().

Here is the caller graph for this function:

15.151.2.11 procedure, private mo_netcdf::ncdataset::getglobalattributei64 ( ) [private]

Definition at line 120 of file mo_netcdf.f90.

Referenced by getattribute().

Here is the caller graph for this function:

15.151.2.12 procedure, private mo_netcdf::ncdataset::getglobalattributei8 ( ) [private]

Definition at line 117 of file mo_netcdf.f90.

Referenced by getattribute().

Here is the caller graph for this function:

15.151.2.13 procedure, public mo_netcdf::ncdataset::getnovariables ( )

Definition at line 104 of file mo_netcdf.f90.

Referenced by mo_netcdf::getnovariables().

Here is the caller graph for this function:

15.151.2.14 procedure, public mo_netcdf::ncdataset::getunlimiteddimension ( )

Return the unlimited dimension of the dataset.

Returns the NcDimension derived type of the unlimited dimension. The program will teminate abruptly if no such dimension exists.

Parameters

in	*character(*) :: name*	

Returns

"logical"

Author

David Schaefer

Date

June 2015

Definition at line 227 of file mo_netcdf.f90.

Referenced by mo_netcdf::getunlimiteddimension().

Here is the caller graph for this function:

15.151.2.15 generic, public mo_netcdf::ncdataset::getvariable ( )

Retrieve NcVariable.

Retrieve the NcVariable derived type for the variable specified by its name. The program will teminate abruptly if no such dimension exists.

Parameters

in	*character(∗) :: name*	

Returns

NcVariable

Author

David Schaefer

Date

June 2015

Definition at line 531 of file mo_netcdf.f90.

References getvariablebyname(), and mo_netcdf::newncdataset().

Here is the call graph for this function:

15.151.2.16 procedure, private mo_netcdf::ncdataset::getvariablebyname ( ) [private]

Definition at line 131 of file mo_netcdf.f90.

Referenced by getvariable(), and mo_netcdf::getvariablebyname().

Here is the caller graph for this function:

15.151.2.17 procedure, public mo_netcdf::ncdataset::getvariableids ( )

Definition at line 105 of file mo_netcdf.f90.

Referenced by mo_netcdf::getvariableids().

Here is the caller graph for this function:

15.151.2.18 procedure, public mo_netcdf::ncdataset::getvariables ( )

Definition at line 106 of file mo_netcdf.f90.

Referenced by mo_netcdf::getvariables().

Here is the caller graph for this function:

15.151.2.19 procedure, public mo_netcdf::ncdataset::hasdimension ( )

Check if dimension exists.

Returns true if a dimension with the given name exists, false otherwise.

Parameters

in	*character(∗) :: name*	

Returns

"logical"

Author

David Schaefer

Date

June 2015

Definition at line 195 of file mo_netcdf.f90.

Referenced by mo_netcdf::hasdimension().

Here is the caller graph for this function:

15.151.2.20 procedure, public mo_netcdf::ncdataset::hasvariable ( )

Check if variable exists.

Returns true if a variable with the given name exists, false otherwise.

Parameters

in	*character(∗) :: name*	

Returns

"logical"

Author

David Schaefer

Date

June 2015

Definition at line 163 of file mo_netcdf.f90.

Referenced by mo_netcdf::hasvariable().

Here is the caller graph for this function:

15.151.2.21 procedure, public mo_netcdf::ncdataset::initncdataset ( )

Definition at line 102 of file mo_netcdf.f90.

15.151.2.22 procedure, public mo_netcdf::ncdataset::isunlimited ( )

Check if the dataset is unlimited.

Returns true if the dataset contains an unlimited dimension, false otherwise.

Returns

"logical"

Author

David Schaefer

Date

June 2015

Definition at line 259 of file mo_netcdf.f90.

References mo_netcdf::isdatasetunlimited().

Referenced by mo_netcdf::ncdimension::isunlimited().

Here is the call graph for this function:

Here is the caller graph for this function:

15.151.2.23 generic, public mo_netcdf::ncdataset::setattribute ( )

Create a new global Attribute.

Create a new global attribute from given name and value. The program will teminate abruptly if the attribute cannot be created.

Parameters

in	*character(∗) :: name*	
in	*character(∗)/integer(i4)/real(sp)/real(dp) :: value*	

Author

David Schaefer

Date

June 2015

Definition at line 359 of file mo_netcdf.f90.

References setglobalattributechar(), setglobalattributef32(), setglobalattributef64(), setglobalattributei16(), setglobalattributei32(), setglobalattributei64(), and setglobalattributei8().

Referenced by mo_netcdf::ncdimension::setattribute().

Here is the call graph for this function:

Here is the caller graph for this function:

15.151.2.24 procedure, public mo_netcdf::ncdataset::setdimension ( )

Create a new dimension.

Create a new dimension of given length. A length < 0 indicates an unlimited dimension. The program will teminate abruptly if the dimension cannot be created.

Parameters

in	*character(∗) :: name*	
in	*integer(i4) :: length*	

Returns

NcDimension

Author

David Schaefer

Date

June 2015

Definition at line 325 of file mo_netcdf.f90.

Referenced by mo_netcdf::setdimension().

Here is the caller graph for this function:

15.151.2.25 procedure, private mo_netcdf::ncdataset::setglobalattributechar ( ) [private]

Definition at line 108 of file mo_netcdf.f90.

Referenced by setattribute().

Here is the caller graph for this function:

15.151.2.26 procedure, private mo_netcdf::ncdataset::setglobalattributef32 ( ) [private]

Definition at line 113 of file mo_netcdf.f90.

Referenced by setattribute().

Here is the caller graph for this function:

15.151.2.27 procedure, private mo_netcdf::ncdataset::setglobalattributef64 ( ) [private]

Definition at line 114 of file mo_netcdf.f90.

Referenced by setattribute().

Here is the caller graph for this function:

15.151.2.28 procedure, private mo_netcdf::ncdataset::setglobalattributei16 ( ) [private]

Definition at line 110 of file mo_netcdf.f90.

Referenced by setattribute().

Here is the caller graph for this function:

15.151.2.29 procedure, private mo_netcdf::ncdataset::setglobalattributei32 ( ) [private]

Definition at line 111 of file mo_netcdf.f90.

Referenced by setattribute().

Here is the caller graph for this function:

15.151.2.30 procedure, private mo_netcdf::ncdataset::setglobalattributei64 ( ) [private]

Definition at line 112 of file mo_netcdf.f90.

Referenced by setattribute().

Here is the caller graph for this function:

15.151.2.31 procedure, private mo_netcdf::ncdataset::setglobalattributei8 ( ) [private]

Definition at line 109 of file mo_netcdf.f90.

Referenced by setattribute().

Here is the caller graph for this function:

15.151.2.32 generic, public mo_netcdf::ncdataset::setvariable ( )

Create a NetCDF variable.

Create a NetCDF Variable with given name, data type and dimensions. All optional arguments to the nf90_def_var function are supported. The program will teminate abruptly if the variable cannot be created. Supported data types and their string encodings: NF90_BYTE -> "i8" NF90_SHORT -> "i16" NF90_INT -> "i32" NF90_INT64 -> "i64" NF90_FLOAT -> "f32" NF90_DOUBLE -> "f64"

Parameters

in	*character(∗) :: name*	
in	*character(3) :: dtype*	
in	*integer(i4)/character(∗)/type(↩ NcDataset) :: dimensions(:)*	
in	*logical :: contiguous*	
in	*integer(i4) :: chunksizes(:)*	
in	*integer(i4) :: deflate_level*	
in	*logical :: shuffle*	
in	*logical :: fletcher32*	
in	*integer(i4) :: endianess*	
in	*integer(i4) :: cache_size*	
in	*integer(i4) :: cache_nelems*	
in	*integer(i4) :: cache_↩ preemption*	

Returns

NcVariable

Author

David Schaefer

Date

June 2015

Definition at line 495 of file mo_netcdf.f90.

References setvariablewithids(), setvariablewithnames(), and setvariablewithtypes().

Here is the call graph for this function:

15.151.2.33 procedure, private mo_netcdf::ncdataset::setvariablewithids ( ) [private]

Definition at line 129 of file mo_netcdf.f90.

Referenced by setvariable(), mo_netcdf::setvariablewithids(), mo_netcdf::setvariablewithnames(), and mo_netcdf↩ ::setvariablewithtypes().

Here is the caller graph for this function:

15.151.2.34 procedure, private mo_netcdf::ncdataset::setvariablewithnames ( ) [private]

Definition at line 128 of file mo_netcdf.f90.

Referenced by setvariable(), and mo_netcdf::setvariablewithnames().

Here is the caller graph for this function:

15.151.2.35 procedure, private mo_netcdf::ncdataset::setvariablewithtypes ( ) [private]

Definition at line 127 of file mo_netcdf.f90.

Referenced by setvariable(), and mo_netcdf::setvariablewithtypes().

Here is the caller graph for this function:

15.151.3 Member Data Documentation

15.151.3.1 character(256) mo_netcdf::ncdataset::dataset

Definition at line 96 of file mo_netcdf.f90.

15.151.3.2 character(1) mo_netcdf::ncdataset::file

Definition at line 97 of file mo_netcdf.f90.

15.151.3.3 character(256) mo_netcdf::ncdataset::filename

Definition at line 96 of file mo_netcdf.f90.

15.151.3.4 character(256) mo_netcdf::ncdataset::fname

Definition at line 96 of file mo_netcdf.f90.

15.151.3.5 integer(i4) mo_netcdf::ncdataset::id

Definition at line 98 of file mo_netcdf.f90.

15.151.3.6 character(1) mo_netcdf::ncdataset::mode

Definition at line 97 of file mo_netcdf.f90.

15.151.3.7 integer(i4) mo_netcdf::ncdataset::netcdf

Definition at line 98 of file mo_netcdf.f90.

15.151.3.8 character(256) mo_netcdf::ncdataset::of

Definition at line 96 of file mo_netcdf.f90.

15.151.3.9 character(1) mo_netcdf::ncdataset::open

Definition at line 97 of file mo_netcdf.f90.

15.151.3.10 character(256) mo_netcdf::ncdataset::opened

Definition at line 96 of file mo_netcdf.f90.

15.151.3.11 character(256) mo_netcdf::ncdataset::the

Definition at line 96 of file mo_netcdf.f90.

The documentation for this interface was generated from the following file:

- mo_netcdf.f90

15.152 mo_netcdf::ncdimension Type Reference

Provides the dimension access functionality.

Collaboration diagram for mo_netcdf::ncdimension:

Public Member Functions

- procedure, public initncvariable
- procedure, public getname => getVariableName
- procedure, public getnodimensions

  Retrieve the number of dimensions.
- procedure, public getdimensions => getVariableDimensions

  Retrieve the variable dimensions.
- procedure, public getshape => getVariableShape

  Retrieve the shape of the variable.
- procedure, public getdtype => getVariableDtype

  Retrieve the variable data type.
- procedure, public hasattribute

  Check if attribute exists.
- procedure, public isunlimited => isUnlimitedVariable

  Check if the variable is unlimited.
- generic, public setdata =>setDataScalarI8,setData1dI8,setData2dI8,setData3dI8,setData4dI8,setData5d↩ I8,setDataScalarI16,setData1dI16,setData2dI16,setData3dI16,setData4dI16,setData5dI16,setDataScalar↩ I32,setData1dI32,setData2dI32,setData3dI32,setData4dI32,setData5dI32,setDataScalarI64,setData1d↩ I64,setData2dI64,setData3dI64,setData4dI64,setData5dI64,setDataScalarF32,setData1dF32,setData2d↩

F32,setData3dF32,setData4dF32,setData5dF32,setDataScalarF64,setData1dF64,setData2dF64,set↩ Data3dF64,setData4dF64,setData5dF64

Write data to variable.

- generic, public getdata =>getDataScalarI8,getData1dI8,getData2dI8,getData3dI8,getData4dI8,getData5d↩ I8,getDataScalarI16,getData1dI16,getData2dI16,getData3dI16,getData4dI16,getData5dI16,getDataScalar↩ I32,getData1dI32,getData2dI32,getData3dI32,getData4dI32,getData5dI32,getDataScalarI64,getData1d↩ I64,getData2dI64,getData3dI64,getData4dI64,getData5dI64,getDataScalarF32,getData1dF32,getData2d↩ F32,getData3dF32,getData4dF32,getData5dF32,getDataScalarF64,getData1dF64,getData2dF64,get↩ Data3dF64,getData4dF64,getData5dF64

  Retrieve data.
- generic, public setfillvalue =>setVariableFillValueI8,setVariableFillValueI16,setVariableFillValueI32,set↩ VariableFillValueI64,setVariableFillValueF32,setVariableFillValueF64

  Set the variable fill value.
- generic, public getfillvalue =>getVariableFillValueI8,getVariableFillValueI16,getVariableFillValueI32,get↩ VariableFillValueI64,getVariableFillValueF32,getVariableFillValueF64

  Retrieve the variable fill value.
- generic, public setattribute =>setVariableAttributeChar,setVariableAttributeI8,setVariableAttributeI16,set↩ VariableAttributeI32,setVariableAttributeI64,setVariableAttributeF32,setVariableAttributeF64

  Create a new variable attribute.
- generic, public getattribute =>getVariableAttributeChar,getVariableAttributeI8,getVariableAttributeI16,get↩ VariableAttributeI32,getVariableAttributeI64,getVariableAttributeF32,getVariableAttributeF64

  Retrieve variable attribute value.

Public Attributes

- integer(i4) id
- integer(i4) the
- integer(i4) netcdf
- integer(i4) dimension
- type(ncdataset) parent
- type(ncdataset) the
- type(ncdataset) dimension
- type(ncdataset) s

Private Member Functions

- procedure, private setvariableattributechar
- procedure, private setvariableattributei8
- procedure, private setvariableattributei16
- procedure, private setvariableattributei32
- procedure, private setvariableattributei64
- procedure, private setvariableattributef32
- procedure, private setvariableattributef64
- procedure, private getvariableattributechar
- procedure, private getvariableattributei8
- procedure, private getvariableattributei16
- procedure, private getvariableattributei32
- procedure, private getvariableattributei64
- procedure, private getvariableattributef32
- procedure, private getvariableattributef64
- procedure, private setdatascalari8
- procedure, private setdata1di8
- procedure, private setdata2di8

- procedure, private setdata3di8
- procedure, private setdata4di8
- procedure, private setdata5di8
- procedure, private setdatascalari16
- procedure, private setdata1di16
- procedure, private setdata2di16
- procedure, private setdata3di16
- procedure, private setdata4di16
- procedure, private setdata5di16
- procedure, private setdatascalari32
- procedure, private setdata1di32
- procedure, private setdata2di32
- procedure, private setdata3di32
- procedure, private setdata4di32
- procedure, private setdata5di32
- procedure, private setdatascalari64
- procedure, private setdata1di64
- procedure, private setdata2di64
- procedure, private setdata3di64
- procedure, private setdata4di64
- procedure, private setdata5di64
- procedure, private setdatascalarf32
- procedure, private setdata1df32
- procedure, private setdata2df32
- procedure, private setdata3df32
- procedure, private setdata4df32
- procedure, private setdata5df32
- procedure, private setdatascalarf64
- procedure, private setdata1df64
- procedure, private setdata2df64
- procedure, private setdata3df64
- procedure, private setdata4df64
- procedure, private setdata5df64
- procedure, private getdatascalari8
- procedure, private getdata1di8
- procedure, private getdata2di8
- procedure, private getdata3di8
- procedure, private getdata4di8
- procedure, private getdata5di8
- procedure, private getdatascalari16
- procedure, private getdata1di16
- procedure, private getdata2di16
- procedure, private getdata3di16
- procedure, private getdata4di16
- procedure, private getdata5di16
- procedure, private getdatascalari32
- procedure, private getdata1di32
- procedure, private getdata2di32
- procedure, private getdata3di32
- procedure, private getdata4di32
- procedure, private getdata5di32
- procedure, private getdatascalari64
- procedure, private getdata1di64
- procedure, private getdata2di64
- procedure, private getdata3di64

- procedure, private getdata4di64
- procedure, private getdata5di64
- procedure, private getdatascalarf32
- procedure, private getdata1df32
- procedure, private getdata2df32
- procedure, private getdata3df32
- procedure, private getdata4df32
- procedure, private getdata5df32
- procedure, private getdatascalarf64
- procedure, private getdata1df64
- procedure, private getdata2df64
- procedure, private getdata3df64
- procedure, private getdata4df64
- procedure, private getdata5df64
- procedure, private setvariablefillvaluei8
- procedure, private setvariablefillvaluei16
- procedure, private setvariablefillvaluei32
- procedure, private setvariablefillvaluei64
- procedure, private setvariablefillvaluef32
- procedure, private setvariablefillvaluef64
- procedure, private getvariablefillvaluei8
- procedure, private getvariablefillvaluei16
- procedure, private getvariablefillvaluei32
- procedure, private getvariablefillvaluei64
- procedure, private getvariablefillvaluef32
- procedure, private getvariablefillvaluef64

15.152.1 Detailed Description

Provides the dimension access functionality.

Bound to this derived type is some necessary inquire functionality. This type is not to be instantiated directly! Use the getDimension/setDimension functions of a NcDataset instance as a "constructor".

Definition at line 572 of file mo_netcdf.f90.

15.152.2 Member Function/Subroutine Documentation

15.152.2.1 generic, public mo_netcdf::ncdimension::getattribute ( )

Retrieve variable attribute value.

Retrieve the value for a variable attribute specified by its name. The program will teminate abruptly if the attribute does not exist.

Parameters

in	*character(∗) :: name*	
out	*character(∗)/integer(i4)/real(sp)/real(dp) :: value*	

Author

David Schaefer

Date

June 2015

Definition at line 1325 of file mo_netcdf.f90.

References mo_netcdf::ncdataset::getattribute(), getvariableattributechar(), getvariableattributef32(), getvariableattributef64(), getvariableattributei16(), getvariableattributei32(), getvariableattributei64(), and getvariableattributei8().

Here is the call graph for this function:

15.152.2.2 generic, public mo_netcdf::ncdimension::getdata ( )

Retrieve data.

Read the data from an optionally given position. All optional arguments to the nf90_get_var function are supported. A read error will result in abrupt program termination.

Parameters

in	*integer(i4) :: start(:), cnt(:), stride(:), map(:)*	
out	*integer(i4)/real(sp)/real(dp), allocatable, dimension(/(:)/(:,:)/(:,:,:)/(:,:,:,:)/(:,:,:,:,:)) :: values*	

Author

David Schaefer

Date

June 2015

Definition at line 1136 of file mo_netcdf.f90.

References getdata1df32(), getdata1df64(), getdata1di16(), getdata1di32(), getdata1di64(), getdata1di8(), getdata2df32(), getdata2df64(), getdata2di16(), getdata2di32(), getdata2di64(), getdata2di8(), getdata3df32(), getdata3df64(), getdata3di16(), getdata3di32(), getdata3di64(), getdata3di8(), getdata4df32(), getdata4df64(), getdata4di16(), getdata4di32(), getdata4di64(), getdata4di8(), getdata5df32(), getdata5df64(), getdata5di16(), getdata5di32(), getdata5di64(), getdata5di8(), getdatascalarf32(), getdatascalarf64(), getdatascalari16(), getdatascalari32(), getdatascalari64(), and getdatascalari8().

Here is the call graph for this function:

15.152.2.3 procedure, private mo_netcdf::ncdimension::getdata1df32 ( ) [private]

Definition at line 807 of file mo_netcdf.f90.

Referenced by getdata().

Here is the caller graph for this function:

15.152.2.4 procedure, private mo_netcdf::ncdimension::getdata1df64 ( ) [private]

Definition at line 813 of file mo_netcdf.f90.

Referenced by getdata().

Here is the caller graph for this function:

15.152.2.5 procedure, private mo_netcdf::ncdimension::getdata1di16 ( ) [private]

Definition at line 789 of file mo_netcdf.f90.

Referenced by getdata().

Here is the caller graph for this function:

15.152.2.6 procedure, private mo_netcdf::ncdimension::getdata1di32 ( ) [private]

Definition at line 795 of file mo_netcdf.f90.

Referenced by getdata().

Here is the caller graph for this function:

15.152.2.7 procedure, private mo_netcdf::ncdimension::getdata1di64 ( ) [private]

Definition at line 801 of file mo_netcdf.f90.

Referenced by getdata().

Here is the caller graph for this function:

15.152.2.8 procedure, private mo_netcdf::ncdimension::getdata1di8 ( ) [private]

Definition at line 783 of file mo_netcdf.f90.

Referenced by getdata().

Here is the caller graph for this function:

15.152.2.9 procedure, private mo_netcdf::ncdimension::getdata2df32 ( ) [private]

Definition at line 808 of file mo_netcdf.f90.

Referenced by getdata().

Here is the caller graph for this function:

15.152.2.10 procedure, private mo_netcdf::ncdimension::getdata2df64 ( ) [private]

Definition at line 814 of file mo_netcdf.f90.

Referenced by getdata().

Here is the caller graph for this function:

15.152.2.11 procedure, private mo_netcdf::ncdimension::getdata2di16 ( ) [private]

Definition at line 790 of file mo_netcdf.f90.

Referenced by getdata().

Here is the caller graph for this function:

15.152.2.12 procedure, private mo_netcdf::ncdimension::getdata2di32 ( ) [private]

Definition at line 796 of file mo_netcdf.f90.

Referenced by getdata().

Here is the caller graph for this function:

15.152.2.13 procedure, private mo_netcdf::ncdimension::getdata2di64 ( ) [private]

Definition at line 802 of file mo_netcdf.f90.

Referenced by getdata().

Here is the caller graph for this function:

15.152.2.14 procedure, private mo_netcdf::ncdimension::getdata2di8 ( ) [private]

Definition at line 784 of file mo_netcdf.f90.

Referenced by getdata().

Here is the caller graph for this function:

15.152.2.15 procedure, private mo_netcdf::ncdimension::getdata3df32 ( ) [private]

Definition at line 809 of file mo_netcdf.f90.

Referenced by getdata().

Here is the caller graph for this function:

15.152.2.16 procedure, private mo_netcdf::ncdimension::getdata3df64 ( ) [private]

Definition at line 815 of file mo_netcdf.f90.

Referenced by getdata().

Here is the caller graph for this function:

15.152.2.17 procedure, private mo_netcdf::ncdimension::getdata3di16 ( ) [private]

Definition at line 791 of file mo_netcdf.f90.

Referenced by getdata().

Here is the caller graph for this function:

15.152.2.18 procedure, private mo_netcdf::ncdimension::getdata3di32 ( ) [private]

Definition at line 797 of file mo_netcdf.f90.

Referenced by getdata().

Here is the caller graph for this function:

15.152.2.19 procedure, private mo_netcdf::ncdimension::getdata3di64 ( ) [private]

Definition at line 803 of file mo_netcdf.f90.

Referenced by getdata().

Here is the caller graph for this function:

15.152.2.20 procedure, private mo_netcdf::ncdimension::getdata3di8 ( ) [private]

Definition at line 785 of file mo_netcdf.f90.

Referenced by getdata().

Here is the caller graph for this function:

15.152.2.21 procedure, private mo_netcdf::ncdimension::getdata4df32 ( ) [private]

Definition at line 810 of file mo_netcdf.f90.

Referenced by getdata().

Here is the caller graph for this function:

15.152.2.22 procedure, private mo_netcdf::ncdimension::getdata4df64 ( ) [private]

Definition at line 816 of file mo_netcdf.f90.

Referenced by getdata().

Here is the caller graph for this function:

15.152.2.23 procedure, private mo_netcdf::ncdimension::getdata4di16 ( ) [private]

Definition at line 792 of file mo_netcdf.f90.

Referenced by getdata().

Here is the caller graph for this function:

15.152.2.24 procedure, private mo_netcdf::ncdimension::getdata4di32 ( ) [private]

Definition at line 798 of file mo_netcdf.f90.

Referenced by getdata().

Here is the caller graph for this function:

15.152.2.25 procedure, private mo_netcdf::ncdimension::getdata4di64 ( ) [private]

Definition at line 804 of file mo_netcdf.f90.

Referenced by getdata().

Here is the caller graph for this function:

15.152.2.26 procedure, private mo_netcdf::ncdimension::getdata4di8 ( ) [private]

Definition at line 786 of file mo_netcdf.f90.

Referenced by getdata().

Here is the caller graph for this function:

15.152.2.27 procedure, private mo_netcdf::ncdimension::getdata5df32 ( ) [private]

Definition at line 811 of file mo_netcdf.f90.

Referenced by getdata().

Here is the caller graph for this function:

15.152.2.28 procedure, private mo_netcdf::ncdimension::getdata5df64 ( ) [private]

Definition at line 817 of file mo_netcdf.f90.

Referenced by getdata().

Here is the caller graph for this function:

15.152.2.29 procedure, private mo_netcdf::ncdimension::getdata5di16 ( ) [private]

Definition at line 793 of file mo_netcdf.f90.

Referenced by getdata().

Here is the caller graph for this function:

15.152.2.30 procedure, private mo_netcdf::ncdimension::getdata5di32 ( ) [private]

Definition at line 799 of file mo_netcdf.f90.

Referenced by getdata().

Here is the caller graph for this function:

15.152.2.31 procedure, private mo_netcdf::ncdimension::getdata5di64 ( ) [private]

Definition at line 805 of file mo_netcdf.f90.

Referenced by getdata().

Here is the caller graph for this function:

15.152.2.32 procedure, private mo_netcdf::ncdimension::getdata5di8 ( ) [private]

Definition at line 787 of file mo_netcdf.f90.

Referenced by getdata().

Here is the caller graph for this function:

15.152.2.33 procedure, private mo_netcdf::ncdimension::getdatascalarf32 ( ) [private]

Definition at line 806 of file mo_netcdf.f90.

Referenced by getdata().

Here is the caller graph for this function:

15.152.2.34 procedure, private mo_netcdf::ncdimension::getdatascalarf64 ( ) [private]

Definition at line 812 of file mo_netcdf.f90.

Referenced by getdata().

Here is the caller graph for this function:

15.152.2.35 procedure, private mo_netcdf::ncdimension::getdatascalari16 ( ) [private]

Definition at line 788 of file mo_netcdf.f90.

Referenced by getdata().

Here is the caller graph for this function:

15.152.2.36 procedure, private mo_netcdf::ncdimension::getdatascalari32 ( ) [private]

Definition at line 794 of file mo_netcdf.f90.

Referenced by getdata().

Here is the caller graph for this function:

15.152.2.37 procedure, private mo_netcdf::ncdimension::getdatascalari64 ( ) [private]

Definition at line 800 of file mo_netcdf.f90.

Referenced by getdata().

Here is the caller graph for this function:

15.152.2.38 procedure, private mo_netcdf::ncdimension::getdatascalari8 ( ) [private]

Definition at line 782 of file mo_netcdf.f90.

Referenced by getdata().

Here is the caller graph for this function:

15.152.2.39 procedure, public mo_netcdf::ncdimension::getdimensions ( )

Retrieve the variable dimensions.

Return the ids of the dimensions associated with variable.

Definition at line 893 of file mo_netcdf.f90.

References mo_netcdf::getvariabledimensions().

Here is the call graph for this function:

15.152.2.40 procedure, public mo_netcdf::ncdimension::getdtype ( )

Retrieve the variable data type.

Return the encoded data type of the variable. Data type encodeings "f32" -> NF90_FLOAT "f64" -> NF90_DOU↩ BLE "i8" -> NF90_BYTE "i16" -> NF90_SHORT "i32" -> NF90_INT "i64" -> NF90_INT64

Definition at line 962 of file mo_netcdf.f90.

References mo_netcdf::getvariabledtype().

Here is the call graph for this function:

15.152.2.41 generic, public mo_netcdf::ncdimension::getfillvalue ( )

Retrieve the variable fill value.

Retrieve the variable fill value or a default value if fill value was not explicitly set. A read error results in abrupt program temination.

Parameters

out	*integer(i4)/real(sp)/real(dp) :: fvalue*	

Author

David Schaefer

Date

June 2015

Definition at line 1246 of file mo_netcdf.f90.

References getvariablefillvaluef32(), getvariablefillvaluef64(), getvariablefillvaluei16(), getvariablefillvaluei32(), getvariablefillvaluei64(), and getvariablefillvaluei8().

Here is the call graph for this function:

15.152.2.42 procedure, public mo_netcdf::ncdimension::getname ( )

Definition at line 727 of file mo_netcdf.f90.

References mo_netcdf::getvariablename().

Here is the call graph for this function:

15.152.2.43 procedure, public mo_netcdf::ncdimension::getnodimensions ( )

Retrieve the number of dimensions.

Return the number of dimensions associated with variable

Definition at line 862 of file mo_netcdf.f90.

Referenced by mo_netcdf::getnodimensions().

Here is the caller graph for this function:

15.152.2.44 procedure, public mo_netcdf::ncdimension::getshape ( )

Retrieve the shape of the variable.

Return the shape of the variable.

Definition at line 924 of file mo_netcdf.f90.

References mo_netcdf::getvariableshape().

Here is the call graph for this function:

15.152.2.45 procedure, private mo_netcdf::ncdimension::getvariableattributechar ( ) [private]

Definition at line 737 of file mo_netcdf.f90.

Referenced by getattribute().

Here is the caller graph for this function:

15.152.2.46 procedure, private mo_netcdf::ncdimension::getvariableattributef32 ( ) [private]

Definition at line 742 of file mo_netcdf.f90.

Referenced by getattribute().

Here is the caller graph for this function:

15.152.2.47 procedure, private mo_netcdf::ncdimension::getvariableattributef64 ( ) [private]

Definition at line 743 of file mo_netcdf.f90.

Referenced by getattribute().

Here is the caller graph for this function:

15.152.2.48 procedure, private mo_netcdf::ncdimension::getvariableattributei16 ( ) [private]

Definition at line 739 of file mo_netcdf.f90.

Referenced by getattribute().

Here is the caller graph for this function:

15.152.2.49 procedure, private mo_netcdf::ncdimension::getvariableattributei32 ( ) [private]

Definition at line 740 of file mo_netcdf.f90.

Referenced by getattribute().

Here is the caller graph for this function:

15.152.2.50 procedure, private mo_netcdf::ncdimension::getvariableattributei64 ( ) [private]

Definition at line 741 of file mo_netcdf.f90.

Referenced by getattribute().

Here is the caller graph for this function:

15.152.2.51 procedure, private mo_netcdf::ncdimension::getvariableattributei8 ( ) [private]

Definition at line 738 of file mo_netcdf.f90.

Referenced by getattribute().

Here is the caller graph for this function:

15.152.2.52 procedure, private mo_netcdf::ncdimension::getvariablefillvaluef32 ( ) [private]

Definition at line 830 of file mo_netcdf.f90.

Referenced by getfillvalue().

Here is the caller graph for this function:

15.152.2.53 procedure, private mo_netcdf::ncdimension::getvariablefillvaluef64 ( ) [private]

Definition at line 831 of file mo_netcdf.f90.

Referenced by getfillvalue().

Here is the caller graph for this function:

15.152.2.54 procedure, private mo_netcdf::ncdimension::getvariablefillvaluei16 ( ) [private]

Definition at line 827 of file mo_netcdf.f90.

Referenced by getfillvalue().

Here is the caller graph for this function:

15.152.2.55 procedure, private mo_netcdf::ncdimension::getvariablefillvaluei32 ( ) [private]

Definition at line 828 of file mo_netcdf.f90.

Referenced by getfillvalue().

Here is the caller graph for this function:

15.152.2.56 procedure, private mo_netcdf::ncdimension::getvariablefillvaluei64 ( ) [private]

Definition at line 829 of file mo_netcdf.f90.

Referenced by getfillvalue().

Here is the caller graph for this function:

15.152.2.57 procedure, private mo_netcdf::ncdimension::getvariablefillvaluei8 ( ) [private]

Definition at line 826 of file mo_netcdf.f90.

Referenced by getfillvalue().

Here is the caller graph for this function:

15.152.2.58 procedure, public mo_netcdf::ncdimension::hasattribute ( )

Check if attribute exists.

Returns true if an attribute with the given name exists, false otherwise.

Definition at line 994 of file mo_netcdf.f90.

Referenced by mo_netcdf::hasattribute().

Here is the caller graph for this function:

15.152.2.59 procedure, public mo_netcdf::ncdimension::initncvariable ( )

Definition at line 725 of file mo_netcdf.f90.

15.152.2.60 procedure, public mo_netcdf::ncdimension::isunlimited ( )

Check if the variable is unlimited.

Returns true if the variable has an unlimited dimension, false otherwise.

Returns

"logical"

Author

David Schaefer

Date

June 2015

Definition at line 1026 of file mo_netcdf.f90.

References mo_netcdf::ncdataset::isunlimited(), and mo_netcdf::isunlimitedvariable().

Here is the call graph for this function:

15.152.2.61 generic, public mo_netcdf::ncdimension::setattribute ( )

Create a new variable attribute.

Create a new variable attribute from given name and value. A write error results in abrupt program temination.

Parameters

in	*character(∗) :: name*	
in	*character(∗)/integer(i4)/real(sp)/real(dp) :: value*	

Author

David Schaefer

Date

June 2015

Definition at line 1285 of file mo_netcdf.f90.

References mo_netcdf::ncdataset::setattribute(), setvariableattributechar(), setvariableattributef32(), setvariableattributef64(), setvariableattributei16(), setvariableattributei32(), setvariableattributei64(), and setvariableattributei8().

Here is the call graph for this function:

15.152.2.62 generic, public mo_netcdf::ncdimension::setdata ( )

Write data to variable.

Write the given data into the variable at an optionally given position. All optional arguments to the nf90_put_var function are supported. A write error will result in abrupt program termination.

Parameters

in	*integer(i4)/real(sp)/real(dp), dimension(/(:)/(:,:)/(:,:,:)/(:,:,:,:)/(:,:,:,:,:)) :: values*	
in	*integer(i4) :: start(:), cnt(:), stride(:), map(:)*	

Author

David Schaefer

Date

June 2015

Definition at line 1063 of file mo_netcdf.f90.

References setdata1df32(), setdata1df64(), setdata1di16(), setdata1di32(), setdata1di64(), setdata1di8(), setdata2df32(), setdata2df64(), setdata2di16(), setdata2di32(), setdata2di64(), setdata2di8(), setdata3df32(), setdata3df64(), setdata3di16(), setdata3di32(), setdata3di64(), setdata3di8(), setdata4df32(), setdata4df64(), setdata4di16(), setdata4di32(), setdata4di64(), setdata4di8(), setdata5df32(), setdata5df64(), setdata5di16(), setdata5di32(), setdata5di64(), setdata5di8(), setdatascalarf32(), setdatascalarf64(), setdatascalari16(), setdatascalari32(), setdatascalari64(), and setdatascalari8().

Here is the call graph for this function:

15.152.2.63 procedure, private mo_netcdf::ncdimension::setdata1df32 ( ) [private]

Definition at line 770 of file mo_netcdf.f90.

Referenced by setdata().

Here is the caller graph for this function:

15.152.2.64 procedure, private mo_netcdf::ncdimension::setdata1df64 ( ) [private]

Definition at line 776 of file mo_netcdf.f90.

Referenced by setdata().

Here is the caller graph for this function:

15.152.2.65 procedure, private mo_netcdf::ncdimension::setdata1di16 ( ) [private]

Definition at line 752 of file mo_netcdf.f90.

Referenced by setdata().

Here is the caller graph for this function:

15.152.2.66 procedure, private mo_netcdf::ncdimension::setdata1di32 ( ) [private]

Definition at line 758 of file mo_netcdf.f90.

Referenced by setdata().

Here is the caller graph for this function:

15.152.2.67 procedure, private mo_netcdf::ncdimension::setdata1di64 ( ) [private]

Definition at line 764 of file mo_netcdf.f90.

Referenced by setdata().

Here is the caller graph for this function:

15.152.2.68 procedure, private mo_netcdf::ncdimension::setdata1di8 ( ) [private]

Definition at line 746 of file mo_netcdf.f90.

Referenced by setdata().

Here is the caller graph for this function:

15.152.2.69 procedure, private mo_netcdf::ncdimension::setdata2df32 ( ) [private]

Definition at line 771 of file mo_netcdf.f90.

Referenced by setdata().

Here is the caller graph for this function:

15.152.2.70 procedure, private mo_netcdf::ncdimension::setdata2df64 ( ) [private]

Definition at line 777 of file mo_netcdf.f90.

Referenced by setdata().

Here is the caller graph for this function:

15.152.2.71 procedure, private mo_netcdf::ncdimension::setdata2di16 ( ) [private]

Definition at line 753 of file mo_netcdf.f90.

Referenced by setdata().

Here is the caller graph for this function:

15.152.2.72 procedure, private mo_netcdf::ncdimension::setdata2di32 ( ) [private]

Definition at line 759 of file mo_netcdf.f90.

Referenced by setdata().

Here is the caller graph for this function:

15.152.2.73 procedure, private mo_netcdf::ncdimension::setdata2di64 ( ) [private]

Definition at line 765 of file mo_netcdf.f90.

Referenced by setdata().

Here is the caller graph for this function:

15.152.2.74 procedure, private mo_netcdf::ncdimension::setdata2di8 ( ) [private]

Definition at line 747 of file mo_netcdf.f90.

Referenced by setdata().

Here is the caller graph for this function:

15.152.2.75 procedure, private mo_netcdf::ncdimension::setdata3df32 ( ) [private]

Definition at line 772 of file mo_netcdf.f90.

Referenced by setdata().

Here is the caller graph for this function:

15.152.2.76 procedure, private mo_netcdf::ncdimension::setdata3df64 ( ) [private]

Definition at line 778 of file mo_netcdf.f90.

Referenced by setdata().

Here is the caller graph for this function:

15.152.2.77 procedure, private mo_netcdf::ncdimension::setdata3di16 ( ) [private]

Definition at line 754 of file mo_netcdf.f90.

Referenced by setdata().

Here is the caller graph for this function:

15.152.2.78 procedure, private mo_netcdf::ncdimension::setdata3di32 ( ) [private]

Definition at line 760 of file mo_netcdf.f90.

Referenced by setdata().

Here is the caller graph for this function:

15.152.2.79 procedure, private mo_netcdf::ncdimension::setdata3di64 ( ) [private]

Definition at line 766 of file mo_netcdf.f90.

Referenced by setdata().

Here is the caller graph for this function:

15.152.2.80 procedure, private mo_netcdf::ncdimension::setdata3di8 ( ) [private]

Definition at line 748 of file mo_netcdf.f90.

Referenced by setdata().

Here is the caller graph for this function:

15.152.2.81 procedure, private mo_netcdf::ncdimension::setdata4df32 ( ) [private]

Definition at line 773 of file mo_netcdf.f90.

Referenced by setdata().

Here is the caller graph for this function:

15.152.2.82 procedure, private mo_netcdf::ncdimension::setdata4df64 ( ) [private]

Definition at line 779 of file mo_netcdf.f90.

Referenced by setdata().

Here is the caller graph for this function:

15.152.2.83 procedure, private mo_netcdf::ncdimension::setdata4di16 ( ) [private]

Definition at line 755 of file mo_netcdf.f90.

Referenced by setdata().

Here is the caller graph for this function:

15.152.2.84 procedure, private mo_netcdf::ncdimension::setdata4di32 ( ) [private]

Definition at line 761 of file mo_netcdf.f90.

Referenced by setdata().

Here is the caller graph for this function:

15.152.2.85 procedure, private mo_netcdf::ncdimension::setdata4di64 ( ) [private]

Definition at line 767 of file mo_netcdf.f90.

Referenced by setdata().

Here is the caller graph for this function:

15.152.2.86 procedure, private mo_netcdf::ncdimension::setdata4di8 ( ) [private]

Definition at line 749 of file mo_netcdf.f90.

Referenced by setdata().

Here is the caller graph for this function:

15.152.2.87 procedure, private mo_netcdf::ncdimension::setdata5df32 ( ) [private]

Definition at line 774 of file mo_netcdf.f90.

Referenced by setdata().

Here is the caller graph for this function:

15.152.2.88 procedure, private mo_netcdf::ncdimension::setdata5df64 ( ) [private]

Definition at line 780 of file mo_netcdf.f90.

Referenced by setdata().

Here is the caller graph for this function:

15.152.2.89 procedure, private mo_netcdf::ncdimension::setdata5di16 ( ) [private]

Definition at line 756 of file mo_netcdf.f90.

Referenced by setdata().

Here is the caller graph for this function:

15.152.2.90 procedure, private mo_netcdf::ncdimension::setdata5di32 ( ) [private]

Definition at line 762 of file mo_netcdf.f90.

Referenced by setdata().

Here is the caller graph for this function:

15.152.2.91 procedure, private mo_netcdf::ncdimension::setdata5di64 ( ) [private]

Definition at line 768 of file mo_netcdf.f90.

Referenced by setdata().

Here is the caller graph for this function:

15.152.2.92 procedure, private mo_netcdf::ncdimension::setdata5di8 ( ) [private]

Definition at line 750 of file mo_netcdf.f90.

Referenced by setdata().

Here is the caller graph for this function:

15.152.2.93 procedure, private mo_netcdf::ncdimension::setdatascalarf32 ( ) [private]

Definition at line 769 of file mo_netcdf.f90.

Referenced by setdata().

Here is the caller graph for this function:

15.152.2.94 procedure, private mo_netcdf::ncdimension::setdatascalarf64 ( ) [private]

Definition at line 775 of file mo_netcdf.f90.

Referenced by setdata().

Here is the caller graph for this function:

15.152.2.95 procedure, private mo_netcdf::ncdimension::setdatascalari16 ( ) [private]

Definition at line 751 of file mo_netcdf.f90.

Referenced by setdata().

Here is the caller graph for this function:

15.152.2.96 procedure, private mo_netcdf::ncdimension::setdatascalari32 ( ) [private]

Definition at line 757 of file mo_netcdf.f90.

Referenced by setdata().

Here is the caller graph for this function:

15.152.2.97 procedure, private mo_netcdf::ncdimension::setdatascalari64 ( ) [private]

Definition at line 763 of file mo_netcdf.f90.

Referenced by setdata().

Here is the caller graph for this function:

15.152.2.98 procedure, private mo_netcdf::ncdimension::setdatascalari8 ( ) [private]

Definition at line 745 of file mo_netcdf.f90.

Referenced by setdata().

Here is the caller graph for this function:

15.152.2.99 generic, public mo_netcdf::ncdimension::setfillvalue ( )

Set the variable fill value.

Define the variable fill value. A write error results in abrupt program temination.

Note

This procedure must be called AFTER the variable was created but BEFORE data is first written.

Parameters

in	*integer(i4)/real(sp)/real(dp) :: fvalue*	

Author

David Schaefer

Date

June 2015

Definition at line 1207 of file mo_netcdf.f90.

References setvariablefillvaluef32(), setvariablefillvaluef64(), setvariablefillvaluei16(), setvariablefillvaluei32(), setvariablefillvaluei64(), and setvariablefillvaluei8().

Here is the call graph for this function:

15.152.2.100 procedure, private mo_netcdf::ncdimension::setvariableattributechar ( ) [private]

Definition at line 729 of file mo_netcdf.f90.

Referenced by setattribute().

Here is the caller graph for this function:

15.152.2.101 procedure, private mo_netcdf::ncdimension::setvariableattributef32 ( ) [private]

Definition at line 734 of file mo_netcdf.f90.

Referenced by setattribute().

Here is the caller graph for this function:

15.152.2.102 procedure, private mo_netcdf::ncdimension::setvariableattributef64 ( ) [private]

Definition at line 735 of file mo_netcdf.f90.

Referenced by setattribute().

Here is the caller graph for this function:

15.152.2.103 procedure, private mo_netcdf::ncdimension::setvariableattributei16 ( ) [private]

Definition at line 731 of file mo_netcdf.f90.

Referenced by setattribute().

Here is the caller graph for this function:

15.152.2.104 procedure, private mo_netcdf::ncdimension::setvariableattributei32 ( ) [private]

Definition at line 732 of file mo_netcdf.f90.

Referenced by setattribute().

Here is the caller graph for this function:

15.152.2.105 procedure, private mo_netcdf::ncdimension::setvariableattributei64 ( ) [private]

Definition at line 733 of file mo_netcdf.f90.

Referenced by setattribute().

Here is the caller graph for this function:

15.152.2.106 procedure, private mo_netcdf::ncdimension::setvariableattributei8 ( ) [private]

Definition at line 730 of file mo_netcdf.f90.

Referenced by setattribute().

Here is the caller graph for this function:

15.152.2.107 procedure, private mo_netcdf::ncdimension::setvariablefillvaluef32 ( ) [private]

Definition at line 823 of file mo_netcdf.f90.

Referenced by setfillvalue().

Here is the caller graph for this function:

15.152.2.108 procedure, private mo_netcdf::ncdimension::setvariablefillvaluef64 ( ) [private]

Definition at line 824 of file mo_netcdf.f90.

Referenced by setfillvalue().

Here is the caller graph for this function:

15.152.2.109 procedure, private mo_netcdf::ncdimension::setvariablefillvaluei16 ( ) [private]

Definition at line 820 of file mo_netcdf.f90.

Referenced by setfillvalue().

Here is the caller graph for this function:

15.152.2.110 procedure, private mo_netcdf::ncdimension::setvariablefillvaluei32 ( ) [private]

Definition at line 821 of file mo_netcdf.f90.

Referenced by setfillvalue().

Here is the caller graph for this function:

15.152.2.111 procedure, private mo_netcdf::ncdimension::setvariablefillvaluei64 ( ) [private]

Definition at line 822 of file mo_netcdf.f90.

Referenced by setfillvalue().

Here is the caller graph for this function:

15.152.2.112 procedure, private mo_netcdf::ncdimension::setvariablefillvaluei8 ( ) [private]

Definition at line 819 of file mo_netcdf.f90.

Referenced by setfillvalue().

Here is the caller graph for this function:

15.152.3 Member Data Documentation

15.152.3.1 integer(i4) mo_netcdf::ncdimension::dimension

Definition at line 574 of file mo_netcdf.f90.

15.152.3.2 type(ncdataset) mo_netcdf::ncdimension::dimension

Definition at line 575 of file mo_netcdf.f90.

15.152.3.3 integer(i4) mo_netcdf::ncdimension::id

Definition at line 574 of file mo_netcdf.f90.

15.152.3.4 integer(i4) mo_netcdf::ncdimension::netcdf

Definition at line 574 of file mo_netcdf.f90.

15.152.3.5 type(ncdataset) mo_netcdf::ncdimension::parent

Definition at line 575 of file mo_netcdf.f90.

15.152.3.6 type(ncdataset) mo_netcdf::ncdimension::s

Definition at line 720 of file mo_netcdf.f90.

15.152.3.7 type(ncdataset) mo_netcdf::ncdimension::the

Definition at line 575 of file mo_netcdf.f90.

15.152.3.8 integer(i4) mo_netcdf::ncdimension::the

Definition at line 574 of file mo_netcdf.f90.

The documentation for this type was generated from the following file:

- mo_netcdf.f90

15.153 mo_netcdf::ncvariable Interface Reference

15.153.1 Detailed Description

Definition at line 1336 of file mo_netcdf.f90.

The documentation for this interface was generated from the following file:

- mo_netcdf.f90

15.154 mo_utils::ne Interface Reference

Public Member Functions

- elemental pure logical function notequal_sp (a, b)
- elemental pure logical function notequal_dp (a, b)

15.154.1 Detailed Description

Definition at line 127 of file mo_utils.f90.

15.154.2 Member Function/Subroutine Documentation

15.154.2.1 elemental pure logical function mo_utils::ne::notequal_dp (real(dp), intent(in) *a*, real(dp), intent(in) *b*)

Definition at line 516 of file mo_utils.f90.

15.154.2.2 elemental pure logical function mo_utils::ne::notequal_sp (real(sp), intent(in) *a*, real(sp), intent(in) *b*)

Definition at line 533 of file mo_utils.f90.

The documentation for this interface was generated from the following file:

- mo_utils.f90

15.155 mo_orderpack::nearless Interface Reference

Public Member Functions

- real(kind=dp) function d_nearless (XVAL)
- real(kind=sp) function r_nearless (XVAL)
- integer(kind=i4) function i_nearless (XVAL)

15.155.1 Detailed Description

Definition at line 339 of file mo_orderpack.f90.

15.155.2 Member Function/Subroutine Documentation

15.155.2.1 real(kind = dp) function mo_orderpack::nearless::d_nearless (real(kind = dp), intent(in) *XVAL*)

Definition at line 10971 of file mo_orderpack.f90.

15.155.2.2 integer(kind = i4) function mo_orderpack::nearless::i_nearless (integer(kind = i4), intent(in) *XVAL*)

Definition at line 10993 of file mo_orderpack.f90.

15.155.2.3 real(kind = sp) function mo_orderpack::nearless::r_nearless (real(kind = sp), intent(in) *XVAL*)

Definition at line 10982 of file mo_orderpack.f90.

The documentation for this interface was generated from the following file:

- mo_orderpack.f90

15.156 mo_spatialsimilarity::nndv Interface Reference

Calculates the number of neighboring dominating values, a measure for spatial dissimilarity.

Public Member Functions

- real(sp) function nndv_sp (mat1, mat2, mask, valid)
- real(dp) function nndv_dp (mat1, mat2, mask, valid)

15.156.1 Detailed Description

Calculates the number of neighboring dominating values, a measure for spatial dissimilarity.

NNDV = 1 - sum(abs(dominating_neighbors(mat1) - dominating_neighbors(mat2))) / count(mask) dominating_↩ neighbors(mat1) = comparison if pixel is larger than its neighbouring values

An array element value is compared with its 8 neighbouring cells to check if these cells are larger than the array element value. The result is a 3x3 matrix in which larger cells are indicated by a true value. For comparison this is done with both input arrays. The resulting array is the sum of substraction of the 3x3 matrices for each of the both arrays. The resulting matrix is afterwards normalized to its available neighbors. Furthermore an average over the entire field is calculated. The valid interval of the values for NNDV is [0..1]. In which 1 indicates full agreement and 0 full dismatching.

```
EXAMPLE:~

mat1 =  | 12 17  1 | , mat2 = |  7  9 12 |
        |  4 10 11 |          | 12 11 11 |
        | 15  2 20 |          |  5 13  7 |
booleans determined for every grid cell following fortran array scrolling
i.e. (/col1_row1, col1_row2, col1_row3, col2_row1, .. ,col3_row3/),(/3,3/)

comp1 = | FFF FFF FTF, FFF FFF FFF, FTT FFT FFF |
        | FFF TFT TTF, TFT TFF FTT, TFF FFT FFF |
        | FFF FFF FFF, TTF TFF TTF, FFF FFF FFF |

comp2 = | FFF FFT FTT, FFT FFT FTT, FFF FFF FFF |
        | FFF FFF FFT, FTF FFT TFF, FFT TFF FFF |
        | FFF TFF TTF, FFF FFF FFF, TTF TFF FFF |
```

```
NNDVMatrix =
abs( count(comp1) - count(comp2) ) = | 1-3, 0-4, 3-0 | = | 2, 4, 3 |
                                     | 4-1, 5-3, 2-2 |   | 3, 2, 0 |
                                     | 0-3, 5-0, 0-3 |   | 3, 5, 3 |

                          DISSIMILAR / VALID NEIGH CELLS
NNDVMatrix / VALID NEIGH CELLS = | 2, 4, 3 | / | 3, 5, 3 |
                                 | 3, 2, 0 |   | 5, 8, 5 |
                                 | 3, 5, 3 |   | 3, 5, 3 |

                               = | 0.66, 0.80, 1.00 |
                                 | 0.60, 0.25, 0.00 |
                                 | 1.0,  1.00, 1.00 |

NNDV = 1 - sum(NNDVMatrix) / count(mask) = 1 - (6.31666666 / 9) = 0.2981
```

If an optinal mask is given, the calculations are over those locations that correspond to true values in the mask. mat1 and mat2 can be single or double precision. The result will have the same numerical precision.

Parameters

in	*real(sp/dp), dimension(:,:) :: mat1*	2D-array with input numbers
in	*real(sp/dp), dimension(:,:) :: mat2*	2D-array with input numbers
in	*logical,dimension(:,:),optinal :: mask*	2D-array of logical values with size(mat1/mat2). If present, only those locations in mat1/mat2 having true values in mask are evaluated.
out	*logical :: valid*	indicates if the function could determine a valid value result can be unvalid if entire mask is .false. for ex. in this case PatternDissim is set to 0 (worst case)

Returns

real(sp/dp) :: NNDV — Number of neighboring dominating values

Note

routine based on algorithm by Luis Samaniego 2009

Author

Matthias Zink

Date

Nov 2012

Definition at line 143 of file mo_spatialsimilarity.f90.

15.156.2 Member Function/Subroutine Documentation

15.156.2.1 real(dp) function mo_spatialsimilarity::nndv::nndv_dp (real(dp), dimension(:, :), intent(in) *mat1,* real(dp), dimension(:, :), intent(in) *mat2,* logical, dimension(:, :), intent(in), optional *mask,* logical, intent(out), optional *valid*)

Definition at line 336 of file mo_spatialsimilarity.f90.

Referenced by mo_spatialsimilarity::nndv_dp().

Here is the caller graph for this function:

15.156.2.2 real(sp) function mo_spatialsimilarity::nndv::nndv_sp (real(sp), dimension(:, :), intent(in) *mat1,* real(sp), dimension(:, :), intent(in) *mat2,* logical, dimension(:, :), intent(in), optional *mask,* logical, intent(out), optional *valid*)

Definition at line 250 of file mo_spatialsimilarity.f90.

Referenced by mo_spatialsimilarity::nndv_sp().

Here is the caller graph for this function:

The documentation for this interface was generated from the following file:

- mo_spatialsimilarity.f90

15.157 mo_utils::notequal Interface Reference

Public Member Functions

- elemental pure logical function notequal_sp (a, b)
- elemental pure logical function notequal_dp (a, b)

15.157.1 Detailed Description

Definition at line 111 of file mo_utils.f90.

15.157.2 Member Function/Subroutine Documentation

15.157.2.1 elemental pure logical function mo_utils::notequal::notequal_dp (real(dp), intent(in) *a*, real(dp), intent(in) *b*)

Definition at line 516 of file mo_utils.f90.

Referenced by mo_utils::notequal_dp().

Here is the caller graph for this function:

15.157.2.2 elemental pure logical function mo_utils::notequal::notequal_sp (real(sp), intent(in) *a*, real(sp), intent(in) *b*)

Definition at line 533 of file mo_utils.f90.

Referenced by mo_utils::notequal_sp().

Here is the caller graph for this function:

The documentation for this interface was generated from the following file:

- mo_utils.f90

15.158 mo_errormeasures::nse Interface Reference

Public Member Functions

- real(sp) function nse_sp_1d (x, y, mask)
- real(dp) function nse_dp_1d (x, y, mask)
- real(dp) function nse_dp_2d (x, y, mask)
- real(sp) function nse_sp_2d (x, y, mask)
- real(sp) function nse_sp_3d (x, y, mask)
- real(dp) function nse_dp_3d (x, y, mask)

15.158.1 Detailed Description

Definition at line 519 of file mo_errormeasures.f90.

15.158.2 Member Function/Subroutine Documentation

15.158.2.1 real(dp) function mo_errormeasures::nse::nse_dp_1d (real(dp), dimension(:), intent(in) *x*, real(dp), dimension(:), intent(in) *y*, logical, dimension(:), intent(in), optional *mask*)

Definition at line 2503 of file mo_errormeasures.f90.

Referenced by mo_errormeasures::nse_dp_1d().

Here is the caller graph for this function:

15.158.2.2 real(dp) function mo_errormeasures::nse::nse_dp_2d (real(dp), dimension(:, :), intent(in) *x*, real(dp), dimension(:, :), intent(in) *y*, logical, dimension(:, :), intent(in), optional *mask*)

Definition at line 2585 of file mo_errormeasures.f90.

Referenced by mo_errormeasures::nse_dp_2d().

Here is the caller graph for this function:

15.158.2.3 real(dp) function mo_errormeasures::nse::nse_dp_3d (real(dp), dimension(:, :, :), intent(in) *x*, real(dp), dimension(:, :, :), intent(in) *y*, logical, dimension(:, :, :), intent(in), optional *mask*)

Definition at line 2666 of file mo_errormeasures.f90.

Referenced by mo_errormeasures::nse_dp_3d().

Here is the caller graph for this function:

15.158.2.4 real(sp) function mo_errormeasures::nse::nse_sp_1d (real(sp), dimension(:), intent(in) *x*, real(sp), dimension(:), intent(in) *y*, logical, dimension(:), intent(in), optional *mask*)

Definition at line 2461 of file mo_errormeasures.f90.

Referenced by mo_errormeasures::nse_sp_1d().

Here is the caller graph for this function:

15.158.2.5 real(sp) function mo_errormeasures::nse::nse_sp_2d (real(sp), dimension(:, :), intent(in) *x*, real(sp), dimension(:, :), intent(in) *y*, logical, dimension(:, :), intent(in), optional *mask*)

Definition at line 2545 of file mo_errormeasures.f90.

Referenced by mo_errormeasures::nse_sp_2d().

Here is the caller graph for this function:

15.158.2.6 real(sp) function mo_errormeasures::nse::nse_sp_3d (real(sp), dimension(:, :, :), intent(in) *x*, real(sp), dimension(:, :, :), intent(in) *y*, logical, dimension(:, :, :), intent(in), optional *mask*)

Definition at line 2625 of file mo_errormeasures.f90.

Referenced by mo_errormeasures::nse_sp_3d().

Here is the caller graph for this function:

The documentation for this interface was generated from the following file:

- mo_errormeasures.f90

15.159 mo_string_utils::num2str Interface Reference

Convert to string.

Public Member Functions

- pure character(len=10) function i42str (nn, form)
- pure character(len=20) function i82str (nn, form)
- pure character(len=32) function sp2str (rr, form)
- pure character(len=32) function dp2str (rr, form)
- pure character(len=10) function log2str (ll, form)

15.159.1 Detailed Description

Convert to string.

Convert a number or logical to a string with an optional format.

Parameters

in	*integer(i4/i8)/real(sp/dp)/logical :: num*	Number or logical
in	*character(len=∗), optinal :: form*	Format string Defaults are: i4 - '(I10)' i8 - '(I20)' sp/dp - '(G32.5)' log - '(L10)'

Returns

character(len=X) :: str — String of formatted input number or logical
Ouput length X is:
i4 - 10
i8 - 20
sp/dp - 32
log - 10

Author

Matthias Cuntz - modified from Echam5, (C) MPI-MET, Hamburg, Germany

Date

Dec 2011

Definition at line 119 of file mo_string_utils.f90.

15.159.2 Member Function/Subroutine Documentation

15.159.2.1 pure character(len = 32) function mo_string_utils::num2str::dp2str (real(dp), intent(in) *rr*, character(len = ∗), intent(in), optional *form*)

Definition at line 847 of file mo_string_utils.f90.

Referenced by mo_string_utils::dp2str().

Here is the caller graph for this function:

15.159.2.2 pure character(len = 10) function mo_string_utils::num2str::i42str (integer(i4), intent(in) *nn*, character(len = ∗), intent(in), optional *form*)

Definition at line 796 of file mo_string_utils.f90.

Referenced by mo_string_utils::i42str().

Here is the caller graph for this function:

15.159.2.3 pure character(len = 20) function mo_string_utils::num2str::i82str (integer(i8), intent(in) *nn*, character(len = ∗), intent(in), optional *form*)

Definition at line 813 of file mo_string_utils.f90.

Referenced by mo_string_utils::i82str().

Here is the caller graph for this function:

15.159.2.4 pure character(len = 10) function mo_string_utils::num2str::log2str (logical, intent(in) *ll*, character(len = ∗), intent(in), optional *form*)

Definition at line 864 of file mo_string_utils.f90.

Referenced by mo_string_utils::log2str().

Here is the caller graph for this function:

15.159.2.5 pure character(len = 32) function mo_string_utils::num2str::sp2str (real(sp), intent(in) *rr*, character(len = ∗), intent(in), optional *form*)

Definition at line 830 of file mo_string_utils.f90.

Referenced by mo_string_utils::sp2str().

Here is the caller graph for this function:

The documentation for this interface was generated from the following file:

- mo_string_utils.f90

15.160 mo_string_utils::numarray2str Interface Reference

Convert to string.

Public Member Functions

- character(len=size(arr)) function i4array2str (arr)

15.160.1 Detailed Description

Convert to string.

Convert a array of numbers or logicals to a string.

Parameters

in	*integer(i4/i8)/real(sp/dp)/logical :: num(:)*	Array of numbers or logicals

Returns

character(len=X) :: str — String of formatted input number or logical

Author

Matthias Cuntz - modified from Echam5, (C) MPI-MET, Hamburg, Germany

Date

Dec 2011

Definition at line 172 of file mo_string_utils.f90.

15.160.2 Member Function/Subroutine Documentation

15.160.2.1 character(len = size(arr)) function mo_string_utils::numarray2str::i4array2str (integer(i4), dimension(:), intent(in) *arr*)

Definition at line 880 of file mo_string_utils.f90.

The documentation for this interface was generated from the following file:

- mo_string_utils.f90

15.161 mo_optimization_utils::objective_interface Interface Reference

Public Member Functions

- real(dp) function objective_interface (parameterset, eval, arg1, arg2, arg3)

15.161.1 Detailed Description

Definition at line 20 of file mo_optimization_utils.f90.

15.161.2 Constructor & Destructor Documentation

15.161.2.1 real(dp) function mo_optimization_utils::objective_interface::objective_interface (real(dp), dimension(:), intent(in) *parameterset,* procedure(eval_interface), intent(in), pointer *eval,* real(dp), intent(in), optional *arg1,* real(dp), intent(out), optional *arg2,* real(dp), intent(out), optional *arg3*)

Definition at line 20 of file mo_optimization_utils.f90.

The documentation for this interface was generated from the following file:

- mo_optimization_utils.f90

15.162 mo_orderpack::omedian Interface Reference

Public Member Functions

- real(kind=dp) function d_median (XDONT)
- real(kind=sp) function r_median (XDONT)
- integer(kind=i4) function i_median (XDONT)

15.162.1 Detailed Description

Definition at line 309 of file mo_orderpack.f90.

15.162.2 Member Function/Subroutine Documentation

15.162.2.1 real(kind = dp) function mo_orderpack::omedian::d_median (real(kind = dp), dimension (:), intent(in) *XDONT*)

Definition at line 3721 of file mo_orderpack.f90.

15.162.2.2 integer(kind = i4) function mo_orderpack::omedian::i_median (integer(kind = i4), dimension (:), intent(in) *XDONT*)

Definition at line 4726 of file mo_orderpack.f90.

15.162.2.3 real(kind = sp) function mo_orderpack::omedian::r_median (real(kind = sp), dimension (:), intent(in) *XDONT*)

Definition at line 4224 of file mo_orderpack.f90.

The documentation for this interface was generated from the following file:

- mo_orderpack.f90

15.163 mo_write_fluxes_states::outputdataset Interface Reference

Collaboration diagram for mo_write_fluxes_states::outputdataset:

Public Member Functions

- procedure, public updatedataset
- procedure, public writetimestep
- procedure, public close

Public Attributes

- integer(i4) ibasin
- integer(i4) basin

- integer(i4) id
- type(ncdataset) nc
- type(ncdataset) ncdataset
- type(ncdataset) to
- type(ncdataset) write
- type(outputvariable),
  dimension(:), allocatable vars
- type(outputvariable), allocatable store
- type(outputvariable), allocatable all
- type(outputvariable),
  dimension(dynamic),
  allocatable created
- type(outputvariable), allocatable variables
- integer(i4) counter = 0
- integer(i4) count
- integer(i4) written
- integer(i4) time
- integer(i4) steps

15.163.1 Detailed Description

Definition at line 44 of file mo_write_fluxes_states.f90.

15.163.2 Member Function/Subroutine Documentation

15.163.2.1 procedure, public mo_write_fluxes_states::outputdataset::close ( )

Definition at line 53 of file mo_write_fluxes_states.f90.

References mo_write_fluxes_states::newoutputdataset().

Here is the call graph for this function:

15.163.2.2 procedure, public mo_write_fluxes_states::outputdataset::updatedataset ( )

Definition at line 51 of file mo_write_fluxes_states.f90.

Referenced by mo_mrm_write_fluxes_states::outputdataset::updatedataset().

Here is the caller graph for this function:

15.163.2.3 procedure, public mo_write_fluxes_states::outputdataset::writetimestep ( )

Definition at line 52 of file mo_write_fluxes_states.f90.

Referenced by mo_mrm_write_fluxes_states::outputdataset::writetimestep().

Here is the caller graph for this function:

15.163.3 Member Data Documentation

15.163.3.1 type(outputvariable), allocatable mo_write_fluxes_states::outputdataset::all

Definition at line 47 of file mo_write_fluxes_states.f90.

15.163.3.2 integer(i4) mo_write_fluxes_states::outputdataset::basin

Definition at line 45 of file mo_write_fluxes_states.f90.

15.163.3.3 integer(i4) mo_write_fluxes_states::outputdataset::count

Definition at line 48 of file mo_write_fluxes_states.f90.

15.163.3.4 integer(i4) mo_write_fluxes_states::outputdataset::counter = 0

Definition at line 48 of file mo_write_fluxes_states.f90.

15.163.3.5 type(outputvariable), dimension (dynamic), allocatable mo_write_fluxes_states::outputdataset::created

Definition at line 47 of file mo_write_fluxes_states.f90.

15.163.3.6 integer(i4) mo_write_fluxes_states::outputdataset::ibasin

Definition at line 45 of file mo_write_fluxes_states.f90.

15.163.3.7 integer(i4) mo_write_fluxes_states::outputdataset::id

Definition at line 45 of file mo_write_fluxes_states.f90.

15.163.3.8 type(ncdataset) mo_write_fluxes_states::outputdataset::nc

Definition at line 46 of file mo_write_fluxes_states.f90.

15.163.3.9 type(ncdataset) mo_write_fluxes_states::outputdataset::ncdataset

Definition at line 46 of file mo_write_fluxes_states.f90.

15.163.3.10 integer(i4) mo_write_fluxes_states::outputdataset::steps

Definition at line 48 of file mo_write_fluxes_states.f90.

15.163.3.11 type(outputvariable), allocatable mo_write_fluxes_states::outputdataset::store

Definition at line 47 of file mo_write_fluxes_states.f90.

15.163.3.12 integer(i4) mo_write_fluxes_states::outputdataset::time

Definition at line 48 of file mo_write_fluxes_states.f90.

15.163.3.13 type(ncdataset) mo_write_fluxes_states::outputdataset::to

Definition at line 46 of file mo_write_fluxes_states.f90.

15.163.3.14 type(outputvariable), allocatable mo_write_fluxes_states::outputdataset::variables

Definition at line 47 of file mo_write_fluxes_states.f90.

15.163.3.15 type(outputvariable), dimension(:), allocatable mo_write_fluxes_states::outputdataset::vars

Definition at line 47 of file mo_write_fluxes_states.f90.

15.163.3.16 type(ncdataset) mo_write_fluxes_states::outputdataset::write

Definition at line 46 of file mo_write_fluxes_states.f90.

15.163.3.17 integer(i4) mo_write_fluxes_states::outputdataset::written

Definition at line 48 of file mo_write_fluxes_states.f90.

The documentation for this interface was generated from the following file:

- mo_write_fluxes_states.f90

15.164 mo_mrm_write_fluxes_states::outputdataset Interface Reference

Collaboration diagram for mo_mrm_write_fluxes_states::outputdataset:

Public Member Functions

- procedure, public updatedataset
- procedure, public writetimestep
- procedure, public close

Public Attributes

- integer(i4) ibasin
- integer(i4) basin

- integer(i4) id
- type(ncdataset) nc
- type(ncdataset) ncdataset
- type(ncdataset) to
- type(ncdataset) write
- type(outputvariable),
  dimension(:), allocatable vars
- type(outputvariable), allocatable store
- type(outputvariable), allocatable all
- type(outputvariable),
  dimension(dynamic),
  allocatable created
- type(outputvariable), allocatable variables
- integer(i4) counter = 0
- integer(i4) count
- integer(i4) written
- integer(i4) time
- integer(i4) steps

15.164.1 Detailed Description

Definition at line 44 of file mo_mrm_write_fluxes_states.f90.

15.164.2 Member Function/Subroutine Documentation

15.164.2.1 procedure, public mo_mrm_write_fluxes_states::outputdataset::close ( )

Definition at line 53 of file mo_mrm_write_fluxes_states.f90.

References mo_write_fluxes_states::newoutputdataset().

Here is the call graph for this function:

15.164.2.2 procedure, public mo_mrm_write_fluxes_states::outputdataset::updatedataset ( )

Definition at line 51 of file mo_mrm_write_fluxes_states.f90.

References mo_write_fluxes_states::outputdataset::updatedataset().

Here is the call graph for this function:

15.164.2.3 procedure, public mo_mrm_write_fluxes_states::outputdataset::writetimestep ( )

Definition at line 52 of file mo_mrm_write_fluxes_states.f90.

References mo_write_fluxes_states::outputdataset::writetimestep().

Here is the call graph for this function:

15.164.3 Member Data Documentation

15.164.3.1 type(outputvariable), allocatable mo_mrm_write_fluxes_states::outputdataset::all

Definition at line 47 of file mo_mrm_write_fluxes_states.f90.

15.164.3.2 integer(i4) mo_mrm_write_fluxes_states::outputdataset::basin

Definition at line 45 of file mo_mrm_write_fluxes_states.f90.

15.164.3.3 integer(i4) mo_mrm_write_fluxes_states::outputdataset::count

Definition at line 48 of file mo_mrm_write_fluxes_states.f90.

15.164.3.4 integer(i4) mo_mrm_write_fluxes_states::outputdataset::counter = 0

Definition at line 48 of file mo_mrm_write_fluxes_states.f90.

15.164.3.5 type(outputvariable), dimension (dynamic), allocatable mo_mrm_write_fluxes_states::outputdataset::created

Definition at line 47 of file mo_mrm_write_fluxes_states.f90.

15.164.3.6 integer(i4) mo_mrm_write_fluxes_states::outputdataset::ibasin

Definition at line 45 of file mo_mrm_write_fluxes_states.f90.

15.164.3.7 integer(i4) mo_mrm_write_fluxes_states::outputdataset::id

Definition at line 45 of file mo_mrm_write_fluxes_states.f90.

15.164.3.8 type(ncdataset) mo_mrm_write_fluxes_states::outputdataset::nc

Definition at line 46 of file mo_mrm_write_fluxes_states.f90.

15.164.3.9 type(ncdataset) mo_mrm_write_fluxes_states::outputdataset::ncdataset

Definition at line 46 of file mo_mrm_write_fluxes_states.f90.

15.164.3.10 integer(i4) mo_mrm_write_fluxes_states::outputdataset::steps

Definition at line 48 of file mo_mrm_write_fluxes_states.f90.

15.164.3.11 type(outputvariable), allocatable mo_mrm_write_fluxes_states::outputdataset::store

Definition at line 47 of file mo_mrm_write_fluxes_states.f90.

15.164.3.12 integer(i4) mo_mrm_write_fluxes_states::outputdataset::time

Definition at line 48 of file mo_mrm_write_fluxes_states.f90.

15.164.3.13 type(ncdataset) mo_mrm_write_fluxes_states::outputdataset::to

Definition at line 46 of file mo_mrm_write_fluxes_states.f90.

15.164.3.14 type(outputvariable), allocatable mo_mrm_write_fluxes_states::outputdataset::variables

Definition at line 47 of file mo_mrm_write_fluxes_states.f90.

15.164.3.15 type(outputvariable), dimension(:), allocatable mo_mrm_write_fluxes_states::outputdataset::vars

Definition at line 47 of file mo_mrm_write_fluxes_states.f90.

15.164.3.16 type(ncdataset) mo_mrm_write_fluxes_states::outputdataset::write

Definition at line 46 of file mo_mrm_write_fluxes_states.f90.

15.164.3.17 integer(i4) mo_mrm_write_fluxes_states::outputdataset::written

Definition at line 48 of file mo_mrm_write_fluxes_states.f90.

The documentation for this interface was generated from the following file:

- mo_mrm_write_fluxes_states.f90

15.165 mo_write_fluxes_states::outputvariable Interface Reference

Collaboration diagram for mo_write_fluxes_states::outputvariable:

Public Member Functions

- procedure, public updatevariable
- procedure, public writevariabletimestep

Public Attributes

- type(ncvariable) nc
- type(ncvariable) ncdataset
- type(ncvariable) which
- type(ncvariable) contains
- type(ncvariable), allocatable the
- type(ncvariable) variable
- logical avg = .false.
- logical average
- logical, dimension(:),
  allocatable, pointer data
- logical before
- logical writing
- logical, dimension(:,:), pointer mask
- logical, pointer to
- logical, pointer reconstruct
- real(dp), dimension(:),
  allocatable, pointer data
- real(dp), allocatable store
- real(dp), allocatable the
- real(dp), allocatable between
- real(dp), allocatable writes
- integer(i4) counter = 0
- integer(i4) count
- integer(i4), allocatable the
- integer(i4) number
- integer(i4) of
- integer(i4), public updatevariable
- integer(i4) calls

15.165.1 Detailed Description

Definition at line 26 of file mo_write_fluxes_states.f90.

15.165.2 Member Function/Subroutine Documentation

15.165.2.1 procedure, public mo_write_fluxes_states::outputvariable::updatevariable ( )

Definition at line 34 of file mo_write_fluxes_states.f90.

Referenced by mo_mrm_write_fluxes_states::updatedataset(), and mo_write_fluxes_states::updatedataset().

Here is the caller graph for this function:

15.165.2.2 procedure, public mo_write_fluxes_states::outputvariable::writevariabletimestep ( )

Definition at line 35 of file mo_write_fluxes_states.f90.

References mo_write_fluxes_states::newoutputvariable().

Referenced by mo_mrm_write_fluxes_states::outputvariable::writevariabletimestep().

Here is the call graph for this function:

Here is the caller graph for this function:

15.165.3 Member Data Documentation

15.165.3.1 logical mo_write_fluxes_states::outputvariable::average

Definition at line 28 of file mo_write_fluxes_states.f90.

15.165.3.2 logical mo_write_fluxes_states::outputvariable::avg = .false.

Definition at line 28 of file mo_write_fluxes_states.f90.

15.165.3.3 logical mo_write_fluxes_states::outputvariable::before

Definition at line 28 of file mo_write_fluxes_states.f90.

15.165.3.4 real(dp), allocatable mo_write_fluxes_states::outputvariable::between

Definition at line 30 of file mo_write_fluxes_states.f90.

15.165.3.5 integer(i4) mo_write_fluxes_states::outputvariable::calls

Definition at line 31 of file mo_write_fluxes_states.f90.

15.165.3.6 type(ncvariable) mo_write_fluxes_states::outputvariable::contains

Definition at line 27 of file mo_write_fluxes_states.f90.

15.165.3.7 integer(i4) mo_write_fluxes_states::outputvariable::count

Definition at line 31 of file mo_write_fluxes_states.f90.

15.165.3.8 integer(i4) mo_write_fluxes_states::outputvariable::counter = 0

Definition at line 31 of file mo_write_fluxes_states.f90.

15.165.3.9 logical, dimension(:), allocatable, pointer mo_write_fluxes_states::outputvariable::data

Definition at line 28 of file mo_write_fluxes_states.f90.

15.165.3.10 real(dp), dimension(:), allocatable, pointer mo_write_fluxes_states::outputvariable::data

Definition at line 30 of file mo_write_fluxes_states.f90.

15.165.3.11 logical, dimension(:, :), pointer mo_write_fluxes_states::outputvariable::mask

Definition at line 29 of file mo_write_fluxes_states.f90.

15.165.3.12 type(ncvariable) mo_write_fluxes_states::outputvariable::nc

Definition at line 27 of file mo_write_fluxes_states.f90.

15.165.3.13 type(ncvariable) mo_write_fluxes_states::outputvariable::ncdataset

Definition at line 27 of file mo_write_fluxes_states.f90.

15.165.3.14 integer(i4) mo_write_fluxes_states::outputvariable::number

Definition at line 31 of file mo_write_fluxes_states.f90.

15.165.3.15 integer(i4) mo_write_fluxes_states::outputvariable::of

Definition at line 31 of file mo_write_fluxes_states.f90.

15.165.3.16 logical, pointer mo_write_fluxes_states::outputvariable::reconstruct

Definition at line 29 of file mo_write_fluxes_states.f90.

15.165.3.17 real(dp), allocatable mo_write_fluxes_states::outputvariable::store

Definition at line 30 of file mo_write_fluxes_states.f90.

15.165.3.18 type(ncvariable), allocatable mo_write_fluxes_states::outputvariable::the

Definition at line 27 of file mo_write_fluxes_states.f90.

15.165.3.19 real(dp), allocatable mo_write_fluxes_states::outputvariable::the

Definition at line 30 of file mo_write_fluxes_states.f90.

15.165.3.20 integer(i4), allocatable mo_write_fluxes_states::outputvariable::the

Definition at line 31 of file mo_write_fluxes_states.f90.

15.165.3.21 logical, pointer mo_write_fluxes_states::outputvariable::to

Definition at line 29 of file mo_write_fluxes_states.f90.

15.165.3.22 integer(i4), public mo_write_fluxes_states::outputvariable::updatevariable

Definition at line 31 of file mo_write_fluxes_states.f90.

15.165.3.23 type(ncvariable) mo_write_fluxes_states::outputvariable::variable

Definition at line 27 of file mo_write_fluxes_states.f90.

15.165.3.24 type(ncvariable) mo_write_fluxes_states::outputvariable::which

Definition at line 27 of file mo_write_fluxes_states.f90.

15.165.3.25 real(dp), allocatable mo_write_fluxes_states::outputvariable::writes

Definition at line 30 of file mo_write_fluxes_states.f90.

15.165.3.26 logical mo_write_fluxes_states::outputvariable::writing

Definition at line 28 of file mo_write_fluxes_states.f90.

The documentation for this interface was generated from the following file:

- mo_write_fluxes_states.f90

15.166 mo_mrm_write_fluxes_states::outputvariable Interface Reference

Collaboration diagram for mo_mrm_write_fluxes_states::outputvariable:

Public Member Functions

- procedure, public updatevariable
- procedure, public writevariabletimestep

Public Attributes

- type(ncvariable) nc
- type(ncvariable) ncdataset
- type(ncvariable) which
- type(ncvariable) contains
- type(ncvariable), allocatable the
- type(ncvariable) variable
- logical avg = .false.
- logical average
- logical, dimension(:),
  allocatable, pointer data
- logical before
- logical writing
- logical, dimension(:,:), pointer mask
- logical, pointer to
- logical, pointer reconstruct
- real(dp), dimension(:),
  allocatable, pointer data
- real(dp), allocatable store
- real(dp), allocatable the
- real(dp), allocatable between
- real(dp), allocatable writes
- integer(i4) counter = 0
- integer(i4) count
- integer(i4), allocatable the
- integer(i4) number
- integer(i4) of
- integer(i4), public updatevariable
- integer(i4) calls

15.166.1 Detailed Description

Definition at line 26 of file mo_mrm_write_fluxes_states.f90.

15.166.2 Member Function/Subroutine Documentation

15.166.2.1 procedure, public mo_mrm_write_fluxes_states::outputvariable::updatevariable ( )

Definition at line 34 of file mo_mrm_write_fluxes_states.f90.

15.166.2.2 procedure, public mo_mrm_write_fluxes_states::outputvariable::writevariabletimestep ( )

Definition at line 35 of file mo_mrm_write_fluxes_states.f90.

References mo_write_fluxes_states::newoutputvariable(), and mo_write_fluxes_states::outputvariable::writevariabletimestep().

Here is the call graph for this function:

15.166.3 Member Data Documentation

15.166.3.1 logical mo_mrm_write_fluxes_states::outputvariable::average

Definition at line 28 of file mo_mrm_write_fluxes_states.f90.

15.166.3.2 logical mo_mrm_write_fluxes_states::outputvariable::avg = .false.

Definition at line 28 of file mo_mrm_write_fluxes_states.f90.

15.166.3.3 logical mo_mrm_write_fluxes_states::outputvariable::before

Definition at line 28 of file mo_mrm_write_fluxes_states.f90.

15.166.3.4 real(dp), allocatable mo_mrm_write_fluxes_states::outputvariable::between

Definition at line 30 of file mo_mrm_write_fluxes_states.f90.

15.166.3.5 integer(i4) mo_mrm_write_fluxes_states::outputvariable::calls

Definition at line 31 of file mo_mrm_write_fluxes_states.f90.

15.166.3.6 type(ncvariable) mo_mrm_write_fluxes_states::outputvariable::contains

Definition at line 27 of file mo_mrm_write_fluxes_states.f90.

15.166.3.7 integer(i4) mo_mrm_write_fluxes_states::outputvariable::count

Definition at line 31 of file mo_mrm_write_fluxes_states.f90.

15.166.3.8 integer(i4) mo_mrm_write_fluxes_states::outputvariable::counter = 0

Definition at line 31 of file mo_mrm_write_fluxes_states.f90.

15.166.3.9 logical, dimension(:), allocatable, pointer mo_mrm_write_fluxes_states::outputvariable::data

Definition at line 28 of file mo_mrm_write_fluxes_states.f90.

15.166.3.10 real(dp), dimension(:), allocatable, pointer mo_mrm_write_fluxes_states::outputvariable::data

Definition at line 30 of file mo_mrm_write_fluxes_states.f90.

15.166.3.11 logical, dimension(:, :), pointer mo_mrm_write_fluxes_states::outputvariable::mask

Definition at line 29 of file mo_mrm_write_fluxes_states.f90.

15.166.3.12 type(ncvariable) mo_mrm_write_fluxes_states::outputvariable::nc

Definition at line 27 of file mo_mrm_write_fluxes_states.f90.

15.166.3.13 type(ncvariable) mo_mrm_write_fluxes_states::outputvariable::ncdataset

Definition at line 27 of file mo_mrm_write_fluxes_states.f90.

15.166.3.14 integer(i4) mo_mrm_write_fluxes_states::outputvariable::number

Definition at line 31 of file mo_mrm_write_fluxes_states.f90.

15.166.3.15 integer(i4) mo_mrm_write_fluxes_states::outputvariable::of

Definition at line 31 of file mo_mrm_write_fluxes_states.f90.

15.166.3.16 logical, pointer mo_mrm_write_fluxes_states::outputvariable::reconstruct

Definition at line 29 of file mo_mrm_write_fluxes_states.f90.

15.166.3.17 real(dp), allocatable mo_mrm_write_fluxes_states::outputvariable::store

Definition at line 30 of file mo_mrm_write_fluxes_states.f90.

15.166.3.18 type(ncvariable), allocatable mo_mrm_write_fluxes_states::outputvariable::the

Definition at line 27 of file mo_mrm_write_fluxes_states.f90.

15.166.3.19 real(dp), allocatable mo_mrm_write_fluxes_states::outputvariable::the

Definition at line 30 of file mo_mrm_write_fluxes_states.f90.

15.166.3.20 integer(i4), allocatable mo_mrm_write_fluxes_states::outputvariable::the

Definition at line 31 of file mo_mrm_write_fluxes_states.f90.

15.166.3.21 logical, pointer mo_mrm_write_fluxes_states::outputvariable::to

Definition at line 29 of file mo_mrm_write_fluxes_states.f90.

15.166.3.22 integer(i4), public mo_mrm_write_fluxes_states::outputvariable::updatevariable

Definition at line 31 of file mo_mrm_write_fluxes_states.f90.

15.166.3.23 type(ncvariable) mo_mrm_write_fluxes_states::outputvariable::variable

Definition at line 27 of file mo_mrm_write_fluxes_states.f90.

15.166.3.24 type(ncvariable) mo_mrm_write_fluxes_states::outputvariable::which

Definition at line 27 of file mo_mrm_write_fluxes_states.f90.

15.166.3.25 real(dp), allocatable mo_mrm_write_fluxes_states::outputvariable::writes

Definition at line 30 of file mo_mrm_write_fluxes_states.f90.

15.166.3.26 logical mo_mrm_write_fluxes_states::outputvariable::writing

Definition at line 28 of file mo_mrm_write_fluxes_states.f90.

The documentation for this interface was generated from the following file:

- mo_mrm_write_fluxes_states.f90

15.167 mo_append::paste Interface Reference

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

Public Member Functions

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

15.167.1 Detailed Description

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

Pastes one input to the columns of another, i.e. append on the second dimension.
The input might be a scalar, a vector or a matrix.
Possibilities are:
(1) paste scalar to one-line matrix
(3) paste vector to a matrix
(5) paste matrix to matrix

Parameters

in	*input2*	values to paste. Can be INTEGER(I4/I8), REAL(SP/DP), or CHARACTE↩ R(len=∗) and also scalar, DIMENSION(:), or DIMENSION(:,:) If not scalar then the rows have to agree with input1
in,out	*allocatable :: input1*	array to be pasted to. Can be INTEGER(I4/I8), REAL(SP/DP), or CHARAC↩ TER(len=∗). Must be DIMENSION(:) or DIMENSION(:,:), and allocatable. If input2 is not scalar then it must be size(input1,1) = size(input2,1).

Author

Juliane Mai

Date

Aug 2012

Definition at line 193 of file mo_append.f90.

15.167.2 Member Function/Subroutine Documentation

15.167.2.1 subroutine mo_append::paste::paste_char_m_m (character(len = ∗), dimension(:, :), intent(inout), allocatable *mat1,* character(len = ∗), dimension(:, :), intent(in) *mat2,* character(len = ∗), intent(in), optional *fill_value*)

Definition at line 1937 of file mo_append.f90.

15.167.2.2 subroutine mo_append::paste::paste_char_m_s (character(len = ∗), dimension(:, :), intent(inout), allocatable *mat1,* character(len = ∗), intent(in) *sca2*)

Definition at line 1845 of file mo_append.f90.

15.167.2.3 subroutine mo_append::paste::paste_char_m_v (character(len = ∗), dimension(:, :), intent(inout), allocatable *mat1,* character(len = ∗), dimension(:), intent(in) *vec2,* character(len = ∗), intent(in), optional *fill_value*)

Definition at line 1879 of file mo_append.f90.

15.167.2.4 subroutine mo_append::paste::paste_dp_m_m (real(dp), dimension(:, :), intent(inout), allocatable *mat1,* real(dp), dimension(:, :), intent(in) *mat2,* real(dp), intent(in), optional *fill_value*)

Definition at line 1787 of file mo_append.f90.

15.167.2.5 subroutine mo_append::paste::paste_dp_m_s (real(dp), dimension(:, :), intent(inout), allocatable *mat1,* real(dp), intent(in) *sca2*)

Definition at line 1695 of file mo_append.f90.

15.167.2.6 subroutine mo_append::paste::paste_dp_m_v (real(dp), dimension(:, :), intent(inout), allocatable *mat1,* real(dp), dimension(:), intent(in) *vec2,* real(dp), intent(in), optional *fill_value*)

Definition at line 1729 of file mo_append.f90.

15.167.2.7 subroutine mo_append::paste::paste_i4_m_m (integer(i4), dimension(:, :), intent(inout), allocatable *mat1,* integer(i4), dimension(:, :), intent(in) *mat2,* integer(i4), intent(in), optional *fill_value*)

Definition at line 1337 of file mo_append.f90.

15.167.2.8 subroutine mo_append::paste::paste_i4_m_s (integer(i4), dimension(:, :), intent(inout), allocatable *mat1,* integer(i4), intent(in) *sca2*)

Definition at line 1245 of file mo_append.f90.

15.167.2.9 subroutine mo_append::paste::paste_i4_m_v (integer(i4), dimension(:, :), intent(inout), allocatable *mat1,* integer(i4), dimension(:), intent(in) *vec2,* integer(i4), intent(in), optional *fill_value*)

Definition at line 1279 of file mo_append.f90.

15.167.2.10 subroutine mo_append::paste::paste_i8_m_m (integer(i8), dimension(:, :), intent(inout), allocatable *mat1,* integer(i8), dimension(:, :), intent(in) *mat2,* integer(i8), intent(in), optional *fill_value*)

Definition at line 1487 of file mo_append.f90.

15.167.2.11 subroutine mo_append::paste::paste_i8_m_s (integer(i8), dimension(:, :), intent(inout), allocatable *mat1,* integer(i8), intent(in) *sca2*)

Definition at line 1395 of file mo_append.f90.

15.167.2.12 subroutine mo_append::paste::paste_i8_m_v (integer(i8), dimension(:, :), intent(inout), allocatable *mat1,* integer(i8), dimension(:), intent(in) *vec2,* integer(i8), intent(in), optional *fill_value*)

Definition at line 1429 of file mo_append.f90.

15.167.2.13 subroutine mo_append::paste::paste_lgt_m_m (logical, dimension(:, :), intent(inout), allocatable *mat1,* logical, dimension(:, :), intent(in) *mat2*)

Definition at line 2069 of file mo_append.f90.

15.167.2.14 subroutine mo_append::paste::paste_lgt_m_s (logical, dimension(:, :), intent(inout), allocatable *mat1,* logical, intent(in) *sca2*)

Definition at line 1995 of file mo_append.f90.

15.167.2.15 subroutine mo_append::paste::paste_lgt_m_v (logical, dimension(:, :), intent(inout), allocatable *mat1,* logical, dimension(:), intent(in) *vec2*)

Definition at line 2029 of file mo_append.f90.

15.167.2.16 subroutine mo_append::paste::paste_sp_m_m (real(sp), dimension(:, :), intent(inout), allocatable *mat1,* real(sp), dimension(:, :), intent(in) *mat2,* real(sp), intent(in), optional *fill_value*)

Definition at line 1637 of file mo_append.f90.

15.167.2.17 subroutine mo_append::paste::paste_sp_m_s (real(sp), dimension(:, :), intent(inout), allocatable *mat1,* real(sp), intent(in) *sca2*)

Definition at line 1545 of file mo_append.f90.

15.167.2.18 subroutine mo_append::paste::paste_sp_m_v (real(sp), dimension(:, :), intent(inout), allocatable *mat1,* real(sp), dimension(:), intent(in) *vec2,* real(sp), intent(in), optional *fill_value*)

Definition at line 1579 of file mo_append.f90.

The documentation for this interface was generated from the following file:

- mo_append.f90

15.168 mo_spatialsimilarity::pd Interface Reference

Calculates pattern dissimilarity (PD) measure.

Public Member Functions

- real(sp) function pd_sp (mat1, mat2, mask, valid)
- real(dp) function pd_dp (mat1, mat2, mask, valid)

15.168.1 Detailed Description

Calculates pattern dissimilarity (PD) measure.

PD = 1 - sum(dissimilarity(mat1, mat2)) / count(mask) dissimilarity(mat1, mat2) = comparison if pixel is larger than its neighbouring values

An array element value is compared with its 8 neighbouring cells to check if these cells are larger than the array element value. The result is a 3x3 matrix in which larger cells are indicated by a true value. For comparison this is done with both input arrays. The resulting array is the sum of xor values of the 3x3 matrices for each of the both arrays. This means only neighbourhood comparisons which are different in the 2 matrices are counted. This resulting matrix is afterwards normalized to its available neighbors. Furthermore an average over the entire field is calculated. The valid interval of the values for PD is [0..1]. In which 1 indicates full agreement and 0 full dismatching.

```
EXAMPLE:~

mat1 =  | 12 17  1 | , mat2 = |  7  9 12 |
        |  4 10 11 |          | 12 11 11 |
        | 15  2 20 |          |  5 13  7 |

booleans determined for every grid cell following fortran array scrolling
i.e. (/col1_row1, col1_row2, col1_row3, col2_row1, .. ,col3_row3/),(/3,3/)

comp1 = | FFF FFF FTF, FFF FFF FFF, FTT FFT FFF |
        | FFF TFT TTF, TFT TFF FTT, TFF FFT FFF |
        | FFF FFF FFF, TTF TFF TTF, FFF FFF FFF |

comp2 = | FFF FFT FTT, FFT FFT FTT, FFF FFF FFF |
        | FFF FFF FFT, FTF FFT TFF, FFT TFF FFF |
        | FFF TFF TTF, FFF FFF FFF, TTF TFF FFF |

xor=neq = | FFF FFT FFT, FFT FFT FTT, FTT FFT FFF |
          | FFF TFT TTT, TTT TFT TTT, TFT TFT FFF |
          | FFF TFF TTF, TTF TFF TTF, TTF TFF FFF |

            DISSIMILAR / VALID NEIGH CELLS
PDMatrix = | 2, 4, 3 | / | 3, 5, 3 | = | 0.66, 0.80, 1.00 |
           | 5, 8, 4 |   | 5, 8, 5 |   | 1.00, 1.00, 0.80 |
           | 3, 5, 3 |   | 3, 5, 3 |   | 1.00, 1.00, 1.00 |
```

```
PD = 1 - sum(PDMatrix) / count(mask) = 1 - (8.2666666 / 9) = 0.08148
```

If an optinal mask is given, the calculations are over those locations that correspond to true values in the mask. mat1 and mat2 can be single or double precision. The result will have the same numerical precision.

Parameters

in	*real(sp/dp), dimension(:,:) :: mat1*	2D-array with input numbers
in	*real(sp/dp), dimension(:,:) :: mat2*	2D-array with input numbers
in	*logical,dimension(:,:),optinal :: mask*	2D-array of logical values with size(mat1/mat2) If present, only those locations in mat1/mat2 having true values in mask are evaluated.
out	*logical,optinal :: valid*	indicates if the function could determine a valid value result can be unvalid if entire mask is .false. for ex. in this case PD is set to 0 (worst case)

Returns

real(sp/dp) :: PD — pattern dissimilarity measure

Author

Matthias Zink and Juliane Mai

Date

Jan 2013

Definition at line 242 of file mo_spatialsimilarity.f90.

15.168.2 Member Function/Subroutine Documentation

15.168.2.1 real(dp) function mo_spatialsimilarity::pd::pd_dp (real(dp), dimension(:, :), intent(in) *mat1,* real(dp), dimension(:, :), intent(in) *mat2,* logical, dimension(:, :), intent(in), optional *mask,* logical, intent(out), optional *valid*)

Definition at line 515 of file mo_spatialsimilarity.f90.

Referenced by mo_spatialsimilarity::pd_dp().

Here is the caller graph for this function:

15.168.2.2 real(sp) function mo_spatialsimilarity::pd::pd_sp (real(sp), dimension(:, :), intent(in) *mat1,* real(sp), dimension(:, :), intent(in) *mat2,* logical, dimension(:, :), intent(in), optional *mask,* logical, intent(out), optional *valid*)

Definition at line 424 of file mo_spatialsimilarity.f90.

Referenced by mo_spatialsimilarity::pd_sp().

Here is the caller graph for this function:

The documentation for this interface was generated from the following file:

- mo_spatialsimilarity.f90

15.169 mo_percentile::percentile Interface Reference

Public Member Functions

- real(sp) function percentile_0d_sp (arrin, k, mask, mode_in)
- real(dp) function percentile_0d_dp (arrin, k, mask, mode_in)
- real(sp) function, dimension(size(k)) percentile_1d_sp (arrin, k, mask, mode_in)
- real(dp) function, dimension(size(k)) percentile_1d_dp (arrin, k, mask, mode_in)

15.169.1 Detailed Description

Definition at line 217 of file mo_percentile.f90.

15.169.2 Member Function/Subroutine Documentation

15.169.2.1 real(dp) function mo_percentile::percentile::percentile_0d_dp (real(dp), dimension(:), intent(in) *arrin,* real(dp), intent(in) *k,* logical, dimension(:), intent(in), optional *mask,* integer(i4), intent(in), optional *mode_in*)

Definition at line 518 of file mo_percentile.f90.

Referenced by mo_percentile::percentile_0d_dp().

Here is the caller graph for this function:

15.169.2.2 real(sp) function mo_percentile::percentile::percentile_0d_sp (real(sp), dimension(:), intent(in) *arrin,* real(sp), intent(in) *k,* logical, dimension(:), intent(in), optional *mask,* integer(i4), intent(in), optional *mode_in*)

Definition at line 629 of file mo_percentile.f90.

Referenced by mo_percentile::percentile_0d_sp().

Here is the caller graph for this function:

15.169.2.3 real(dp) function, dimension(size(k)) mo_percentile::percentile::percentile_1d_dp (real(dp), dimension(:), intent(in) *arrin,* real(dp), dimension(:), intent(in) *k,* logical, dimension(:), intent(in), optional *mask,* integer(i4), intent(in), optional *mode_in*)

Definition at line 740 of file mo_percentile.f90.

Referenced by mo_percentile::percentile_1d_dp().

Here is the caller graph for this function:

15.169.2.4 real(sp) function, dimension(size(k)) mo_percentile::percentile::percentile_1d_sp (real(sp), dimension(:), intent(in) *arrin*, real(sp), dimension(:), intent(in) *k*, logical, dimension(:), intent(in), optional *mask*, integer(i4), intent(in), optional *mode_in*)

Definition at line 860 of file mo_percentile.f90.

Referenced by mo_percentile::percentile_1d_sp().

Here is the caller graph for this function:

The documentation for this interface was generated from the following file:

- mo_percentile.f90

15.170 mo_common_variables::period Type Reference

Collaboration diagram for mo_common_variables::period:

Public Attributes

- integer(i4) dstart
- integer(i4) mstart
- integer(i4) ystart
- integer(i4) dend
- integer(i4) mend
- integer(i4) yend
- integer(i4) julstart
- integer(i4) julend
- integer(i4) nobs

15.170.1 Detailed Description

Definition at line 58 of file mo_common_variables.f90.

15.170.2 Member Data Documentation

15.170.2.1 integer(i4) mo_common_variables::period::dend

Definition at line 62 of file mo_common_variables.f90.

15.170.2.2 integer(i4) mo_common_variables::period::dstart

Definition at line 59 of file mo_common_variables.f90.

15.170.2.3 integer(i4) mo_common_variables::period::julend

Definition at line 66 of file mo_common_variables.f90.

15.170.2.4 integer(i4) mo_common_variables::period::julstart

Definition at line 65 of file mo_common_variables.f90.

15.170.2.5 integer(i4) mo_common_variables::period::mend

Definition at line 63 of file mo_common_variables.f90.

15.170.2.6 integer(i4) mo_common_variables::period::mstart

Definition at line 60 of file mo_common_variables.f90.

15.170.2.7 integer(i4) mo_common_variables::period::nobs

Definition at line 67 of file mo_common_variables.f90.

15.170.2.8 integer(i4) mo_common_variables::period::yend

Definition at line 64 of file mo_common_variables.f90.

15.170.2.9 integer(i4) mo_common_variables::period::ystart

Definition at line 61 of file mo_common_variables.f90.

The documentation for this type was generated from the following file:

- mo_common_variables.f90

15.171 mo_percentile::qmedian Interface Reference

Public Member Functions

- real(sp) function qmedian_sp (dat)
- real(dp) function qmedian_dp (dat)

15.171.1 Detailed Description

Definition at line 267 of file mo_percentile.f90.

15.171.2 Member Function/Subroutine Documentation

15.171.2.1 real(dp) function mo_percentile::qmedian::qmedian_dp (real(dp), dimension(:), intent(inout) *dat*)

Definition at line 981 of file mo_percentile.f90.

Referenced by mo_percentile::qmedian_dp().

Here is the caller graph for this function:

15.171.2.2 real(sp) function mo_percentile::qmedian::qmedian_sp (real(sp), dimension(:), intent(inout) *dat*)

Definition at line 1028 of file mo_percentile.f90.

Referenced by mo_percentile::qmedian_sp().

Here is the caller graph for this function:

The documentation for this interface was generated from the following file:

- mo_percentile.f90

15.172 mo_orderpack::rapknr Interface Reference

Public Member Functions

- subroutine d_rapknr (XDONT, IRNGT, NORD)
- subroutine r_rapknr (XDONT, IRNGT, NORD)
- subroutine i_rapknr (XDONT, IRNGT, NORD)

15.172.1 Detailed Description

Definition at line 321 of file mo_orderpack.f90.

15.172.2 Member Function/Subroutine Documentation

15.172.2.1 subroutine mo_orderpack::rapknr::d_rapknr (real(kind = dp), dimension (:), intent(in) *XDONT,* integer(kind = i4), dimension (:), intent(out) *IRNGT,* integer(kind = i4), intent(in) *NORD*)

Definition at line 6296 of file mo_orderpack.f90.

15.172.2.2 subroutine mo_orderpack::rapknr::i_rapknr (integer(kind = i4), dimension (:), intent(in) *XDONT,* integer(kind = i4), dimension (:), intent(out) *IRNGT,* integer(kind = i4), intent(in) *NORD*)

Definition at line 7339 of file mo_orderpack.f90.

15.172.2.3 subroutine mo_orderpack::rapknr::r_rapknr (real(kind = sp), dimension (:), intent(in) *XDONT,* integer(kind = i4), dimension (:), intent(out) *IRNGT,* integer(kind = i4), intent(in) *NORD*)

Definition at line 6817 of file mo_orderpack.f90.

The documentation for this interface was generated from the following file:

- mo_orderpack.f90

15.173 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.

15.173.1 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 51 of file mo_read_spatial_data.f90.

15.173.2 Member Function/Subroutine Documentation

15.173.2.1 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 93 of file mo_read_spatial_data.f90.

15.173.2.2 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 230 of file mo_read_spatial_data.f90.

The documentation for this interface was generated from the following file:

- mo_read_spatial_data.f90

15.174 mo_corr::realft Interface Reference

Private Member Functions

- subroutine realft_sp (data, isign, zdata)
- subroutine realft_dp (data, isign, zdata)

15.174.1 Detailed Description

Definition at line 386 of file mo_corr.f90.

15.174.2 Member Function/Subroutine Documentation

15.174.2.1 subroutine mo_corr::realft::realft_dp (real(dp), dimension(:), intent(inout) *data*, integer(i4), intent(in) *isign*, complex(dpc), dimension(:), optional, target *zdata*) [private]

Definition at line 1251 of file mo_corr.f90.

15.174.2.2 subroutine mo_corr::realft::realft_sp (real(sp), dimension(:), intent(inout) *data*, integer(i4), intent(in) *isign*, complex(spc), dimension(:), optional, target *zdata*) [private]

Definition at line 1311 of file mo_corr.f90.

The documentation for this interface was generated from the following file:

- mo_corr.f90

15.175 mo_orderpack::refpar Interface Reference

Public Member Functions

- subroutine d_refpar (XDONT, IRNGT, NORD)
- subroutine r_refpar (XDONT, IRNGT, NORD)
- subroutine i_refpar (XDONT, IRNGT, NORD)

15.175.1 Detailed Description

Definition at line 324 of file mo_orderpack.f90.

15.175.2 Member Function/Subroutine Documentation

15.175.2.1 subroutine mo_orderpack::refpar::d_refpar (real(kind = dp), dimension (:), intent(in) *XDONT,* integer(kind = i4), dimension (:), intent(out) *IRNGT,* integer(kind = i4), intent(in) *NORD*)

Definition at line 7861 of file mo_orderpack.f90.

15.175.2.2 subroutine mo_orderpack::refpar::i_refpar (integer(kind = i4), dimension (:), intent(in) *XDONT,* integer(kind = i4), dimension (:), intent(out) *IRNGT,* integer(kind = i4), intent(in) *NORD*)

Definition at line 8095 of file mo_orderpack.f90.

15.175.2.3 subroutine mo_orderpack::refpar::r_refpar (real(kind = sp), dimension (:), intent(in) *XDONT,* integer(kind = i4), dimension (:), intent(out) *IRNGT,* integer(kind = i4), intent(in) *NORD*)

Definition at line 7978 of file mo_orderpack.f90.

The documentation for this interface was generated from the following file:

- mo_orderpack.f90

15.176 mo_orderpack::refsor Interface Reference

Public Member Functions

- subroutine d_refsor (XDONT)
- subroutine r_refsor (XDONT)
- subroutine i_refsor (XDONT)

15.176.1 Detailed Description

Definition at line 327 of file mo_orderpack.f90.

15.176.2 Member Function/Subroutine Documentation

15.176.2.1 subroutine mo_orderpack::refsor::d_refsor (real(kind = dp), dimension (:), intent(inout) *XDONT*)

Definition at line 8211 of file mo_orderpack.f90.

15.176.2.2 subroutine mo_orderpack::refsor::i_refsor (integer(kind = i4), dimension (:), intent(inout) *XDONT*)

Definition at line 8427 of file mo_orderpack.f90.

15.176.2.3 subroutine mo_orderpack::refsor::r_refsor (real(kind = sp), dimension (:), intent(inout) *XDONT*)

Definition at line 8319 of file mo_orderpack.f90.

The documentation for this interface was generated from the following file:

- mo_orderpack.f90

15.177 mo_orderpack::rinpar Interface Reference

Public Member Functions

- subroutine d_rinpar (XDONT, IRNGT, NORD)
- subroutine r_rinpar (XDONT, IRNGT, NORD)
- subroutine i_rinpar (XDONT, IRNGT, NORD)

15.177.1 Detailed Description

Definition at line 330 of file mo_orderpack.f90.

15.177.2 Member Function/Subroutine Documentation

15.177.2.1 subroutine mo_orderpack::rinpar::d_rinpar (real(kind = dp), dimension (:), intent(in) *XDONT,* integer(kind = i4), dimension (:), intent(out) *IRNGT,* integer(kind = i4), intent(in) *NORD*)

Definition at line 8535 of file mo_orderpack.f90.

15.177.2.2 subroutine mo_orderpack::rinpar::i_rinpar (integer(kind = i4), dimension (:), intent(in) *XDONT,* integer(kind = i4), dimension (:), intent(out) *IRNGT,* integer(kind = i4), intent(in) *NORD*)

Definition at line 8625 of file mo_orderpack.f90.

15.177.2.3 subroutine mo_orderpack::rinpar::r_rinpar (real(kind = sp), dimension (:), intent(in) *XDONT,* integer(kind = i4), dimension (:), intent(out) *IRNGT,* integer(kind = i4), intent(in) *NORD*)

Definition at line 8580 of file mo_orderpack.f90.

The documentation for this interface was generated from the following file:

- mo_orderpack.f90

15.178 mo_errormeasures::rmse Interface Reference

Public Member Functions

- real(sp) function rmse_sp_1d (x, y, mask)
- real(dp) function rmse_dp_1d (x, y, mask)
- real(sp) function rmse_sp_2d (x, y, mask)
- real(dp) function rmse_dp_2d (x, y, mask)
- real(sp) function rmse_sp_3d (x, y, mask)
- real(dp) function rmse_dp_3d (x, y, mask)

15.178.1 Detailed Description

Definition at line 702 of file mo_errormeasures.f90.

15.178.2 Member Function/Subroutine Documentation

15.178.2.1 real(dp) function mo_errormeasures::rmse::rmse_dp_1d (real(dp), dimension(:), intent(in) *x*, real(dp), dimension(:), intent(in) *y*, logical, dimension(:), intent(in), optional *mask*)

Definition at line 3174 of file mo_errormeasures.f90.

Referenced by mo_errormeasures::rmse_dp_1d().

Here is the caller graph for this function:

15.178.2.2 real(dp) function mo_errormeasures::rmse::rmse_dp_2d (real(dp), dimension(:, :), intent(in) *x*, real(dp), dimension(:, :), intent(in) *y*, logical, dimension(:, :), intent(in), optional *mask*)

Definition at line 3240 of file mo_errormeasures.f90.

Referenced by mo_errormeasures::rmse_dp_2d().

Here is the caller graph for this function:

15.178.2.3 real(dp) function mo_errormeasures::rmse::rmse_dp_3d (real(dp), dimension(:, :, :), intent(in) *x*, real(dp), dimension(:, :, :), intent(in) *y*, logical, dimension(:, :, :), intent(in), optional *mask*)

Definition at line 3307 of file mo_errormeasures.f90.

Referenced by mo_errormeasures::rmse_dp_3d().

Here is the caller graph for this function:

15.178.2.4 real(sp) function mo_errormeasures::rmse::rmse_sp_1d (real(sp), dimension(:), intent(in) *x*, real(sp), dimension(:), intent(in) *y*, logical, dimension(:), intent(in), optional *mask*)

Definition at line 3141 of file mo_errormeasures.f90.

Referenced by mo_errormeasures::rmse_sp_1d().

Here is the caller graph for this function:

15.178.2.5 real(sp) function mo_errormeasures::rmse::rmse_sp_2d (real(sp), dimension(:, :), intent(in) *x*, real(sp), dimension(:, :), intent(in) *y*, logical, dimension(:, :), intent(in), optional *mask*)

Definition at line 3207 of file mo_errormeasures.f90.

Referenced by mo_errormeasures::rmse_sp_2d().

Here is the caller graph for this function:

15.178.2.6 real(sp) function mo_errormeasures::rmse::rmse_sp_3d (real(sp), dimension(:, :, :), intent(in) *x*, real(sp), dimension(:, :, :), intent(in) *y*, logical, dimension(:, :, :), intent(in), optional *mask*)

Definition at line 3273 of file mo_errormeasures.f90.

Referenced by mo_errormeasures::rmse_sp_3d().

Here is the caller graph for this function:

The documentation for this interface was generated from the following file:

- mo_errormeasures.f90

15.179 mo_orderpack::rnkpar Interface Reference

Public Member Functions

- subroutine d_rnkpar (XDONT, IRNGT, NORD)
- subroutine r_rnkpar (XDONT, IRNGT, NORD)
- subroutine i_rnkpar (XDONT, IRNGT, NORD)

15.179.1 Detailed Description

Definition at line 333 of file mo_orderpack.f90.

15.179.2 Member Function/Subroutine Documentation

15.179.2.1 subroutine mo_orderpack::rnkpar::d_rnkpar (real(kind = dp), dimension (:), intent(in) *XDONT*, integer(kind = i4), dimension (:), intent(out) *IRNGT*, integer(kind = i4), intent(in) *NORD*)

Definition at line 8670 of file mo_orderpack.f90.

15.179.2.2 subroutine mo_orderpack::rnkpar::i_rnkpar (integer(kind = i4), dimension (:), intent(in) *XDONT*, integer(kind = i4), dimension (:), intent(out) *IRNGT*, integer(kind = i4), intent(in) *NORD*)

Definition at line 9710 of file mo_orderpack.f90.

15.179.2.3 subroutine mo_orderpack::rnkpar::r_rnkpar (real(kind = sp), dimension (:), intent(in) *XDONT*, integer(kind = i4), dimension (:), intent(out) *IRNGT*, integer(kind = i4), intent(in) *NORD*)

Definition at line 9190 of file mo_orderpack.f90.

The documentation for this interface was generated from the following file:

- mo_orderpack.f90

15.180 mo_errormeasures::sae Interface Reference

Public Member Functions

- real(sp) function sae_sp_1d (x, y, mask)
- real(dp) function sae_dp_1d (x, y, mask)
- real(sp) function sae_sp_2d (x, y, mask)
- real(dp) function sae_dp_2d (x, y, mask)
- real(sp) function sae_sp_3d (x, y, mask)
- real(dp) function sae_dp_3d (x, y, mask)

15.180.1 Detailed Description

Definition at line 580 of file mo_errormeasures.f90.

15.180.2 Member Function/Subroutine Documentation

15.180.2.1 real(dp) function mo_errormeasures::sae::sae_dp_1d (real(dp), dimension(:), intent(in) *x*, real(dp), dimension(:), intent(in) *y*, logical, dimension(:), intent(in), optional *mask*)

Definition at line 2744 of file mo_errormeasures.f90.

Referenced by mo_errormeasures::mae_dp_1d(), mo_errormeasures::sae_dp_1d(), mo_errormeasures::sae_dp↩_2d(), and mo_errormeasures::sae_dp_3d().

Here is the caller graph for this function:

15.180.2.2 real(dp) function mo_errormeasures::sae::sae_dp_2d (real(dp), dimension(:, :), intent(in) *x*, real(dp), dimension(:, :), intent(in) *y*, logical, dimension(:, :), intent(in), optional *mask*)

Definition at line 2814 of file mo_errormeasures.f90.

Referenced by mo_errormeasures::mae_dp_2d(), and mo_errormeasures::sae_dp_2d().

Here is the caller graph for this function:

15.180.2.3 real(dp) function mo_errormeasures::sae::sae_dp_3d (real(dp), dimension(:, :, :), intent(in) *x*, real(dp), dimension(:, :, :), intent(in) *y*, logical, dimension(:, :, :), intent(in), optional *mask*)

Definition at line 2888 of file mo_errormeasures.f90.

Referenced by mo_errormeasures::mae_dp_3d(), and mo_errormeasures::sae_dp_3d().

Here is the caller graph for this function:

15.180.2.4 real(sp) function mo_errormeasures::sae::sae_sp_1d (real(sp), dimension(:), intent(in) *x*, real(sp), dimension(:), intent(in) *y*, logical, dimension(:), intent(in), optional *mask*)

Definition at line 2710 of file mo_errormeasures.f90.

Referenced by mo_errormeasures::mae_sp_1d(), mo_errormeasures::sae_sp_1d(), mo_errormeasures::sae_sp↩_2d(), and mo_errormeasures::sae_sp_3d().

Here is the caller graph for this function:

15.180.2.5 real(sp) function mo_errormeasures::sae::sae_sp_2d (real(sp), dimension(:, :), intent(in) *x*, real(sp), dimension(:, :), intent(in) *y*, logical, dimension(:, :), intent(in), optional *mask*)

Definition at line 2778 of file mo_errormeasures.f90.

Referenced by mo_errormeasures::mae_sp_2d(), and mo_errormeasures::sae_sp_2d().

Here is the caller graph for this function:

15.180.2.6 real(sp) function mo_errormeasures::sae::sae_sp_3d (real(sp), dimension(:, :, :), intent(in) *x*, real(sp), dimension(:, :, :), intent(in) *y*, logical, dimension(:, :, :), intent(in), optional *mask*)

Definition at line 2850 of file mo_errormeasures.f90.

Referenced by mo_errormeasures::mae_sp_3d(), and mo_errormeasures::sae_sp_3d().

Here is the caller graph for this function:

The documentation for this interface was generated from the following file:

- mo_errormeasures.f90

15.181 mo_julian::setcalendar Interface Reference

Private Member Functions

- subroutine setcalendarinteger (selector)

  Set module private variable calendar.
- subroutine setcalendarstring (selector)

  Set module private variable calendar.

15.181.1 Detailed Description

Definition at line 70 of file mo_julian.f90.

15.181.2 Member Function/Subroutine Documentation

15.181.2.1 subroutine mo_julian::setcalendar::setcalendarinteger (integer(i4), intent(in) *selector*) [private]

Set module private variable calendar.

Parameters

in	*integer(i4) :: selector*	{1|2|3}

Author

Written, David Schaefer

Date

Jan 2015

Definition at line 157 of file mo_julian.f90.

Referenced by mo_julian::setcalendarstring().

Here is the caller graph for this function:

15.181.2.2 subroutine mo_julian::setcalendar::setcalendarstring (character(*), intent(in) *selector*) [private]

Set module private variable calendar.

Parameters

in	*character(len=*) :: selector*	{"julian"|"365day"|"360day"}

Author

Written, David Schaefer

Date

Jan 2015

Definition at line 109 of file mo_julian.f90.

The documentation for this interface was generated from the following file:

- mo_julian.f90

15.182 mo_moment::skewness Interface Reference

Public Member Functions

- real(sp) function skewness_sp (dat, mask)
- real(dp) function skewness_dp (dat, mask)

15.182.1 Detailed Description

Definition at line 725 of file mo_moment.f90.

15.182.2 Member Function/Subroutine Documentation

15.182.2.1 real(dp) function mo_moment::skewness::skewness_dp (real(dp), dimension(:), intent(in) *dat,* logical, dimension(:), intent(in), optional *mask*)

Definition at line 1805 of file mo_moment.f90.

Referenced by mo_moment::skewness_dp().

Here is the caller graph for this function:

15.182.2.2 real(sp) function mo_moment::skewness::skewness_sp (real(sp), dimension(:), intent(in) *dat,* logical, dimension(:), intent(in), optional *mask*)

Definition at line 1847 of file mo_moment.f90.

Referenced by mo_moment::skewness_sp().

Here is the caller graph for this function:

The documentation for this interface was generated from the following file:

- mo_moment.f90

15.183 mo_orderpack::sort Interface Reference

Unconditional ranking.

Public Member Functions

- subroutine d_refsor (XDONT)
- subroutine r_refsor (XDONT)
- subroutine i_refsor (XDONT)

15.183.1 Detailed Description

Unconditional ranking.

Subroutine MRGRNK (XVALT, IMULT) Ranks array XVALT into index array IRNGT, using merge-sort For performance reasons, the first 2 passes are taken out of the standard loop, and use dedicated coding.

Subroutine MRGREF (XVALT, IRNGT) Ranks array XVALT into index array IRNGT, using merge-sort This version is not optimized for performance, and is thus not as difficult to read as the previous one.

Partial ranking

Subroutine RNKPAR (XVALT, IRNGT, NORD) Ranks partially XVALT by IRNGT, up to order NORD (refined for speed) This routine uses a pivoting strategy such as the one of finding the median based on the quicksort algorithm, but we skew the pivot choice to try to bring it to NORD as fast as possible. It uses 2 temporary arrays, one where it stores the indices of the values smaller than the pivot, and the other for the indices of values larger than the pivot that we might still need later on. It iterates until it can bring the number of values in ILOWT to exactly NORD, and then uses an insertion sort to rank this set, since it is supposedly small.

Subroutine RAPKNR (XVALT, IRNGT, NORD) Same as RNKPAR, but in decreasing order (RAPKNR = RNKPAR spelt backwards).

Subroutine REFPAR (XVALT, IRNGT, NORD) Ranks partially XVALT by IRNGT, up to order NORD This version is not optimized for performance, and is thus not as difficult to read as some other ones. It uses a pivoting strategy such as the one of finding the median based on the quicksort algorithm. It uses a temporary array, where it stores the partially ranked indices of the values. It iterates until it can bring the number of values lower than the pivot to exactly NORD, and then uses an insertion sort to rank this set, since it is supposedly small.

Subroutine RINPAR (XVALT, IRNGT, NORD) Ranks partially XVALT by IRNGT, up to order NORD This version is not optimized for performance, and is thus not as difficult to read as some other ones. It uses insertion sort, limiting insertion to the first NORD values. It does not use any work array and is faster when NORD is very small (2-5), but worst case behavior (intially inverse sorted) can easily happen. In many cases, the refined quicksort method is faster.

Integer Function INDNTH (XVALT, NORD) Returns the index of the NORDth value of XVALT (in increasing order) This routine uses a pivoting strategy such as the one of finding the median based on the quicksort algorithm, but we skew the pivot choice to try to bring it to NORD as fast as possible. It uses 2 temporary arrays, one where it stores the indices of the values smaller than the pivot, and the other for the indices of values larger than the pivot that we might still need later on. It iterates until it can bring the number of values in ILOWT to exactly NORD, and then takes out the original index of the maximum value in this set.

Subroutine INDMED (XVALT, INDM) Returns the index of the median (((Size(XVALT)+1))/2th value) of XVALT This routine uses the recursive procedure described in Knuth, The Art of Computer Programming, vol. 3, 5.3.3 - This procedure is linear in time, and does not require to be able to interpolate in the set as the one used in INDNT↩H. It also has better worst case behavior than INDNTH, but is about 10% slower in average for random uniformly distributed values.

Note that in Orderpack 1.0, this routine was a Function procedure, and is now changed to a Subroutine.

Unique ranking

Subroutine UNIRNK (XVALT, IRNGT, NUNI) Ranks an array, removing duplicate entries (uses merge sort). The routine is similar to pure merge-sort ranking, but on the last pass, it discards indices that correspond to duplicate entries. For performance reasons, the first 2 passes are taken out of the standard loop, and use dedicated coding.

Subroutine UNIPAR (XVALT, IRNGT, NORD) Ranks partially XVALT by IRNGT, up to order NORD at most, removing duplicate entries This routine uses a pivoting strategy such as the one of finding the median based on the quicksort algorithm, but we skew the pivot choice to try to bring it to NORD as quickly as possible. It uses 2 temporary arrays, one where it stores the indices of the values smaller than the pivot, and the other for the indices of values larger than the pivot that we might still need later on. It iterates until it can bring the number of values in ILOWT to exactly NORD, and then uses an insertion sort to rank this set, since it is supposedly small. At all times, the NORD first values in ILOWT correspond to distinct values of the input array.

Subroutine UNIINV (XVALT, IGOEST) Inverse ranking of an array, with removal of duplicate entries The routine is similar to pure merge-sort ranking, but on the last pass, it sets indices in IGOEST to the rank of the original value in an ordered set with duplicates removed. For performance reasons, the first 2 passes are taken out of the standard

loop, and use dedicated coding.

Subroutine MULCNT (XVALT, IMULT) Gives, for each array value, its multiplicity The number of times that a value appears in the array is computed by using inverse ranking, counting for each rank the number of values that “collide” to this rank, and returning this sum to the locations in the original set. Uses subroutine UNIINV.

Random permutation: an interesting use of ranking

A variation of the following problem was raised on the internet sci.math.num-analysis news group: Given an array, I would like to find a random permutation of this array that I could control with a `nearbyness″ parameter so that elements stay close to their initial locations. The`nearbyness” parameter ranges from 0 to 1, with 0 such that no element moves from its initial location, and 1 such that the permutation is fully random.

Subroutine CTRPER (XVALT, PCLS) Permute array XVALT randomly, but leaving elements close to their initial locations The routine takes the 1...size(XVALT) index array as real values, takes a combination of these values and of random values as a perturbation of the index array, and sorts the initial set according to the ranks of these perturbated indices. The relative proportion of initial order and random order is 1-PCLS / PCLS, thus when PCLS = 0, there is no change in the order whereas the new order is fully random when PCLS = 1. Uses subroutine MRGRNK.

The above solution found another application when I was asked the following question: I am given two arrays, representing parents’ incomes and their children’s incomes, but I do not know which parents correspond to which children. I know from an independent source the value of the correlation coefficient between the incomes of the parents and of their children. I would like to pair the elements of these arrays so that the given correlation coefficient is attained, i.e. to reconstruct a realistic dataset, though very likely not to be the true one.

Program GIVCOR Given two arrays of equal length of unordered values, find a "matching value" in the second array for each value in the first so that the global correlation coefficient reaches exactly a given target The routine first sorts the two arrays, so as to get the match of maximum possible correlation. It then iterates, applying the random permutation algorithm of controlled disorder ctrper to the second array. When the resulting correlation goes beyond (lower than) the target correlation, one steps back and reduces the disorder parameter of the permutation. When the resulting correlation lies between the current one and the target, one replaces the array with the newly permuted one. When the resulting correlation increases from the current value, one increases the disorder parameter. That way, the target correlation is approached from above, by a controlled increase in randomness. Since full randomness leads to zero correlation, the iterations meet the desired coefficient at some point. It may be noted that there could be some cases when one would get stuck in a sort of local minimum, where local perturbations cannot further reduce the correlation and where global ones lead to overpass the target. It seems easier to restart the program with a different seed when this occurs than to design an avoidance scheme. Also, should a negative correlation be desired, the program should be modified to start with one array in reverse order with respect to the other, i.e. coorelation as close to -1 as possible.

Sorting

Full sorting

Subroutine INSSOR (XVALT) Sorts XVALT into increasing order (Insertion sort) This subroutine uses insertion sort. It does not use any work array and is faster when XVALT is of very small size (< 20), or already almost sorted, but worst case behavior (intially inverse sorted) can easily happen. In most cases, the quicksort or merge sort method is faster.

Subroutine REFSOR (XVALT) Sorts XVALT into increasing order (Quick sort) This version is not optimized for performance, and is thus not as difficult to read as some other ones. This subroutine uses quicksort in a recursive implementation, and insertion sort for the last steps with small subsets. It does not use any work array

Partial sorting

Subroutine INSPAR (XVALT, NORD) Sorts partially XVALT, bringing the NORD lowest values at the begining of the array. This subroutine uses insertion sort, limiting insertion to the first NORD values. It does not use any work array and is faster when NORD is very small (2-5), but worst case behavior can happen fairly probably (initially inverse sorted). In many cases, the refined quicksort method is faster.

Function FNDNTH (XVALT, NORD) Finds out and returns the NORDth value in XVALT (ascending order) This subroutine uses insertion sort, limiting insertion to the first NORD values, and even less when one can know that the value that is considered will not be the NORDth. It uses only a work array of size NORD and is faster when

NORD is very small (2-5), but worst case behavior can happen fairly probably (initially inverse sorted). In many cases, the refined quicksort method implemented by VALNTH / INDNTH is faster, though much more difficult to read and understand.

Function VALNTH (XVALT, NORD) Finds out and returns the NORDth value in XVALT (ascending order) This subroutine simply calls INDNTH.

Function VALMED (XVALT) Finds out and returns the median (((Size(XVALT)+1))/2th value) of XVALT This routine uses the recursive procedure described in Knuth, The Art of Computer Programming, vol. 3, 5.3.3 - This procedure is linear in time, and does not require to be able to interpolate in the set as the one used in VALNTH/INDNTH. It also has better worst case behavior than VALNTH/INDNTH, and is about 20% faster in average for random uniformly distributed values.

Function OMEDIAN (XVALT) It is a modified version of VALMED that provides the average between the two middle values in the case Size(XVALT) is even.

Unique sorting

Subroutine UNISTA (XVALT, NUNI) Removes duplicates from an array This subroutine uses merge sort unique inverse ranking. It leaves in the initial set only those entries that are unique, packing the array, and leaving the order of the retained values unchanged.

Definition at line 284 of file mo_orderpack.f90.

15.183.2 Member Function/Subroutine Documentation

15.183.2.1 subroutine mo_orderpack::sort::d_refsor (real(kind = dp), dimension (:), intent(inout) *XDONT*)

Definition at line 8211 of file mo_orderpack.f90.

15.183.2.2 subroutine mo_orderpack::sort::i_refsor (integer(kind = i4), dimension (:), intent(inout) *XDONT*)

Definition at line 8427 of file mo_orderpack.f90.

15.183.2.3 subroutine mo_orderpack::sort::r_refsor (real(kind = sp), dimension (:), intent(inout) *XDONT*)

Definition at line 8319 of file mo_orderpack.f90.

The documentation for this interface was generated from the following file:

- mo_orderpack.f90

15.184 mo_orderpack::sort_index Interface Reference

Public Member Functions

- integer(i4) function,
  dimension(size(arr)) sort_index_dp (arr)
- integer(i4) function,
  dimension(size(arr)) sort_index_sp (arr)
- integer(i4) function,
  dimension(size(arr)) sort_index_i4 (arr)

15.184.1 Detailed Description

Definition at line 287 of file mo_orderpack.f90.

15.184.2 Member Function/Subroutine Documentation

15.184.2.1 integer(i4) function, dimension(size(arr)) mo_orderpack::sort_index::sort_index_dp (real(dp), dimension(:), intent(in) *arr*)

Definition at line 391 of file mo_orderpack.f90.

Referenced by mo_orderpack::sort_index_dp().

Here is the caller graph for this function:

15.184.2.2 integer(i4) function, dimension(size(arr)) mo_orderpack::sort_index::sort_index_i4 (integer(i4), dimension(:), intent(in) *arr*)

Definition at line 413 of file mo_orderpack.f90.

Referenced by mo_orderpack::sort_index_i4().

Here is the caller graph for this function:

15.184.2.3 integer(i4) function, dimension(size(arr)) mo_orderpack::sort_index::sort_index_sp (real(sp), dimension(:), intent(in) *arr*)

Definition at line 402 of file mo_orderpack.f90.

Referenced by mo_orderpack::sort_index_sp().

Here is the caller graph for this function:

The documentation for this interface was generated from the following file:

- mo_orderpack.f90

15.185 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)

15.185.1 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.

15.185.2 Member Function/Subroutine Documentation

15.185.2.1 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.

15.185.2.2 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.

The documentation for this interface was generated from the following file:

- mo_spatial_agg_disagg_forcing.f90

15.186 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)

15.186.1 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.

15.186.2 Member Function/Subroutine Documentation

15.186.2.1 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.

15.186.2.2 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.

The documentation for this interface was generated from the following file:

- mo_spatial_agg_disagg_forcing.f90

15.187 mo_utils::special_value Interface Reference

Special IEEE values.

Public Member Functions

- real(sp) function special_value_sp (x, ieee)
- real(dp) function special_value_dp (x, ieee)

15.187.1 Detailed Description

Special IEEE values.

Returns special IEEE values such as Infinity or Not-a-Number.
Wraps to function ieee_value of the intrinsic module ieee_arithmetic but gives alternatives for gfortran, which does not provide ieee_arithmetic.
Quiet and signaling NaN are the same in case of gfortran;
also denormal values are the same as inf.

Current special values are:
IEEE_SIGNALING_NAN
IEEE_QUIET_NAN
IEEE_NEGATIVE_INF
IEEE_POSITIVE_INF
IEEE_NEGATIVE_DENORMAL
IEEE_POSITIVE_DENORMAL
IEEE_NEGATIVE_NORMAL
IEEE_POSITIVE_NORMAL
IEEE_NEGATIVE_ZERO
IEEE_POSITIVE_ZERO

Parameters

in	*real(sp/dp) :: x*	dummy for kind of output

in	*"character(le=∗)*	:: name ieee signal nanme

Returns

real(sp/dp) :: special_value — IEEE special value
IEEE_SIGNALING_NAN
IEEE_QUIET_NAN (==IEEE_SIGNALING_NAN for gfortran)
IEEE_NEGATIVE_INF
IEEE_POSITIVE_INF
IEEE_NEGATIVE_DENORMAL (==-0.0 for gfortran)
IEEE_POSITIVE_DENORMAL (==0.0 for gfortran)
IEEE_NEGATIVE_NORMAL (==-1.0 for gfortran)
IEEE_POSITIVE_NORMAL (==1.0 for gfortran)
IEEE_NEGATIVE_ZERO
IEEE_POSITIVE_ZERO

Authors

Matthias Cuntz

Date

Mar 2015

Definition at line 405 of file mo_utils.f90.

15.187.2 Member Function/Subroutine Documentation

15.187.2.1 real(dp) function mo_utils::special_value::special_value_dp (real(dp), intent(in) *x*, character(len = ∗), intent(in) *ieee*)

Definition at line 861 of file mo_utils.f90.

Referenced by mo_utils::special_value_dp().

Here is the caller graph for this function:

15.187.2.2 real(sp) function mo_utils::special_value::special_value_sp (real(sp), intent(in) *x*, character(len = ∗), intent(in) *ieee*)

Definition at line 948 of file mo_utils.f90.

Referenced by mo_utils::special_value_sp().

Here is the caller graph for this function:

The documentation for this interface was generated from the following file:

- mo_utils.f90

15.188 mo_kind::sprs2_dp Type Reference

Double Precision Numerical Recipes types for sparse arrays.

Collaboration diagram for mo_kind::sprs2_dp:

Public Attributes

- integer(i4) n
- integer(i4) len
- real(dp), dimension(:), pointer val
- integer(i4), dimension(:), pointer irow
- integer(i4), dimension(:), pointer jcol

15.188.1 Detailed Description

Double Precision Numerical Recipes types for sparse arrays.

Definition at line 100 of file mo_kind.f90.

15.188.2 Member Data Documentation

15.188.2.1 integer(i4), dimension(:), pointer mo_kind::sprs2_dp::irow

Definition at line 103 of file mo_kind.f90.

15.188.2.2 integer(i4), dimension(:), pointer mo_kind::sprs2_dp::jcol

Definition at line 104 of file mo_kind.f90.

15.188.2.3 integer(i4) mo_kind::sprs2_dp::len

Definition at line 101 of file mo_kind.f90.

15.188.2.4 integer(i4) mo_kind::sprs2_dp::n

Definition at line 101 of file mo_kind.f90.

15.188.2.5 real(dp), dimension(:), pointer mo_kind::sprs2_dp::val

Definition at line 102 of file mo_kind.f90.

The documentation for this type was generated from the following file:

- mo_kind.f90

15.189 mo_kind::sprs2_sp Type Reference

Single Precision Numerical Recipes types for sparse arrays.

Collaboration diagram for mo_kind::sprs2_sp:

Public Attributes

- integer(i4) n
- integer(i4) len
- real(sp), dimension(:), pointer val
- integer(i4), dimension(:), pointer irow
- integer(i4), dimension(:), pointer jcol

15.189.1 Detailed Description

Single Precision Numerical Recipes types for sparse arrays.

Definition at line 93 of file mo_kind.f90.

15.189.2 Member Data Documentation

15.189.2.1 integer(i4), dimension(:), pointer mo_kind::sprs2_sp::irow

Definition at line 96 of file mo_kind.f90.

15.189.2.2 integer(i4), dimension(:), pointer mo_kind::sprs2_sp::jcol

Definition at line 97 of file mo_kind.f90.

15.189.2.3 integer(i4) mo_kind::sprs2_sp::len

Definition at line 94 of file mo_kind.f90.

15.189.2.4 integer(i4) mo_kind::sprs2_sp::n

Definition at line 94 of file mo_kind.f90.

15.189.2.5 real(sp), dimension(:), pointer mo_kind::sprs2_sp::val

Definition at line 95 of file mo_kind.f90.

The documentation for this type was generated from the following file:

- mo_kind.f90

15.190 mo_errormeasures::sse Interface Reference

Public Member Functions

- real(sp) function sse_sp_1d (x, y, mask)
- real(dp) function sse_dp_1d (x, y, mask)
- real(sp) function sse_sp_2d (x, y, mask)
- real(dp) function sse_dp_2d (x, y, mask)
- real(sp) function sse_sp_3d (x, y, mask)
- real(dp) function sse_dp_3d (x, y, mask)

15.190.1 Detailed Description

Definition at line 641 of file mo_errormeasures.f90.

15.190.2 Member Function/Subroutine Documentation

15.190.2.1 real(dp) function mo_errormeasures::sse::sse_dp_1d (real(dp), dimension(:), intent(in) *x*, real(dp), dimension(:), intent(in) *y*, logical, dimension(:), intent(in), optional *mask*)

Definition at line 2962 of file mo_errormeasures.f90.

Referenced by mo_errormeasures::mse_dp_1d(), mo_errormeasures::sse_dp_1d(), mo_errormeasures::sse_dp↩_2d(), and mo_errormeasures::sse_dp_3d().

Here is the caller graph for this function:

15.190.2.2 real(dp) function mo_errormeasures::sse::sse_dp_2d (real(dp), dimension(:, :), intent(in) *x*, real(dp), dimension(:, :), intent(in) *y*, logical, dimension(:, :), intent(in), optional *mask*)

Definition at line 3030 of file mo_errormeasures.f90.

Referenced by mo_errormeasures::mse_dp_2d(), and mo_errormeasures::sse_dp_2d().

Here is the caller graph for this function:

15.190.2.3 real(dp) function mo_errormeasures::sse::sse_dp_3d (real(dp), dimension(:, :, :), intent(in) *x*, real(dp), dimension(:, :, :), intent(in) *y*, logical, dimension(:, :, :), intent(in), optional *mask*)

Definition at line 3102 of file mo_errormeasures.f90.

Referenced by mo_errormeasures::mse_dp_3d(), and mo_errormeasures::sse_dp_3d().

Here is the caller graph for this function:

15.190.2.4 real(sp) function mo_errormeasures::sse::sse_sp_1d (real(sp), dimension(:), intent(in) *x*, real(sp), dimension(:), intent(in) *y*, logical, dimension(:), intent(in), optional *mask*)

Definition at line 2928 of file mo_errormeasures.f90.

Referenced by mo_errormeasures::mse_sp_1d(), mo_errormeasures::sse_sp_1d(), mo_errormeasures::sse_sp↩_2d(), and mo_errormeasures::sse_sp_3d().

Here is the caller graph for this function:

15.190.2.5 real(sp) function mo_errormeasures::sse::sse_sp_2d (real(sp), dimension(:, :), intent(in) *x*, real(sp), dimension(:, :), intent(in) *y*, logical, dimension(:, :), intent(in), optional *mask*)

Definition at line 2995 of file mo_errormeasures.f90.

Referenced by mo_errormeasures::mse_sp_2d(), and mo_errormeasures::sse_sp_2d().

Here is the caller graph for this function:

15.190.2.6 real(sp) function mo_errormeasures::sse::sse_sp_3d (real(sp), dimension(:, :, :), intent(in) *x*, real(sp), dimension(:, :, :), intent(in) *y*, logical, dimension(:, :, :), intent(in), optional *mask*)

Definition at line 3065 of file mo_errormeasures.f90.

Referenced by mo_errormeasures::mse_sp_3d(), and mo_errormeasures::sse_sp_3d().

Here is the caller graph for this function:

The documentation for this interface was generated from the following file:

- mo_errormeasures.f90

15.191 mo_standard_score::standard_score Interface Reference

Calculates the standard score / normalization (anomaly) / z-score.

Public Member Functions

- real(sp) function, dimension(size(data, dim=1)) standard_score_sp (data, mask)
- real(dp) function, dimension(size(data, dim=1)) standard_score_dp (data, mask)

15.191.1 Detailed Description

Calculates the standard score / normalization (anomaly) / z-score.

In statistics, the standard score is the (signed) number of standard deviations an observation or datum is above the mean. Thus, a positive standard score indicates a datum above the mean, while a negative standard score indicates a datum below the mean. It is a dimensionless quantity obtained by subtracting the population mean from an individual raw score and then dividing the difference by the population standard deviation. This conversion process is called standardizing or normalizing (however, "normalizing" can refer to many types of ratios).
Standard scores are also called z-values, z-scores, normal scores, and standardized variables; the use of "Z" is because the normal distribution is also known as the "Z distribution". They are most frequently used to compare a sample to a standard normal deviate, though they can be defined without assumptions of normality (Wikipedia, May 2015).

$$standard_score = \frac{x - \mu_x}{\sigma_x}$$

where μ_x is the mean of a population x and σ_x its standard deviation.

If an optinal mask is given, the calculations are over those locations that correspond to true values in the mask. data can be single or double precision. The result will have the same numerical precision.

Parameters

in	*real(sp/dp), dimension(:) :: data*	data to calculate the standard score for
in	*logical, dimension(↩ :),optinal :: mask*	indication which cells to use for calculation If present, only those locations in mask having true values in mask are evaluated.

Returns

real(sp/dp) :: standard_score — standard score / normalization (anomaly) / z-score

Note

Richard J. Larsen and Morris L. Marx (2000) An Introduction to Mathematical Statistics and Its Applications, Third Edition, ISBN 0-13-922303-7. p. 282.

Author

Matthias Zink

Date

May 2015

Definition at line 114 of file mo_standard_score.f90.

15.191.2 Member Function/Subroutine Documentation

15.191.2.1 real(dp) function, dimension(size(data, dim = 1)) mo_standard_score::standard_score::standard_score_dp (real(dp), dimension(:), intent(in) *data*, logical, dimension(:), intent(in), optional *mask*)

Definition at line 233 of file mo_standard_score.f90.

Referenced by mo_standard_score::standard_score_dp().

Here is the caller graph for this function:

15.191.2.2 real(sp) function, dimension(size(data, dim = 1)) mo_standard_score::standard_score::standard_score_sp (real(sp), dimension(:), intent(in) *data*, logical, dimension(:), intent(in), optional *mask*)

Definition at line 204 of file mo_standard_score.f90.

Referenced by mo_standard_score::standard_score_sp().

Here is the caller graph for this function:

The documentation for this interface was generated from the following file:

- mo_standard_score.f90

15.192 mo_moment::stddev Interface Reference

Public Member Functions

- real(sp) function stddev_sp (dat, mask)
- real(dp) function stddev_dp (dat, mask)

15.192.1 Detailed Description

Definition at line 780 of file mo_moment.f90.

15.192.2 Member Function/Subroutine Documentation

15.192.2.1 real(dp) function mo_moment::stddev::stddev_dp (real(dp), dimension(:), intent(in) *dat*, logical, dimension(:), intent(in), optional *mask*)

Definition at line 1730 of file mo_moment.f90.

Referenced by mo_moment::stddev_dp().

Here is the caller graph for this function:

15.192.2.2 real(sp) function mo_moment::stddev::stddev_sp (real(sp), dimension(:), intent(in) *dat,* logical, dimension(:), intent(in), optional *mask*)

Definition at line 1767 of file mo_moment.f90.

Referenced by mo_moment::stddev_sp().

Here is the caller graph for this function:

The documentation for this interface was generated from the following file:

- mo_moment.f90

15.193 mo_corr::swap Interface Reference

15.193.1 Detailed Description

Definition at line 389 of file mo_corr.f90.

The documentation for this interface was generated from the following file:

- mo_corr.f90

15.194 mo_utils::swap Interface Reference

Swap to values or two elements in array.

Public Member Functions

- elemental pure subroutine swap_xy_dp (x, y)
- elemental pure subroutine swap_xy_sp (x, y)
- elemental pure subroutine swap_xy_i4 (x, y)
- subroutine swap_vec_dp (x, i1, i2)
- subroutine swap_vec_sp (x, i1, i2)
- subroutine swap_vec_i4 (x, i1, i2)

15.194.1 Detailed Description

Swap to values or two elements in array.

Swaps either two entities, i.e. scalars, vectors, matrices, or two elements in a vector. The call is either
call swap(x,y)
or
call swap(vec,i,j)

Parameters

in	*integer(i4) :: i*	Index of first element to be swapped with second [case swap(vec,i,j)]
in	*integer(i4) :: j*	Index of second element to be swapped with first [case swap(vec,i,j)]
in,out	*real(sp/dp/i4) :: x[(:,...)]*	First scalar or array to swap with second [case swap(x,y)]
in,out	*real(sp/dp/i4) :: y[(:[,:])]*	Second scalar or array to swap with first [case swap(x,y)]
in,out	*real(sp/dp/i4) :: x(:)*	Vector of which to elements are swapped [case swap(vec,i,j)]

Author

Matthias Cuntz

Date

May 2014

Definition at line 324 of file mo_utils.f90.

15.194.2 Member Function/Subroutine Documentation

15.194.2.1 subroutine mo_utils::swap::swap_vec_dp (real(dp), dimension(:), intent(inout) *x*, integer(i4), intent(in) *i1*, integer(i4), intent(in) *i2*)

Definition at line 811 of file mo_utils.f90.

15.194.2.2 subroutine mo_utils::swap::swap_vec_i4 (integer(i4), dimension(:), intent(inout) *x*, integer(i4), intent(in) *i1*, integer(i4), intent(in) *i2*)

Definition at line 843 of file mo_utils.f90.

15.194.2.3 subroutine mo_utils::swap::swap_vec_sp (real(sp), dimension(:), intent(inout) *x*, integer(i4), intent(in) *i1*, integer(i4), intent(in) *i2*)

Definition at line 827 of file mo_utils.f90.

15.194.2.4 elemental pure subroutine mo_utils::swap::swap_xy_dp (real(dp), intent(inout) *x*, real(dp), intent(inout) *y*)

Definition at line 765 of file mo_utils.f90.

15.194.2.5 elemental pure subroutine mo_utils::swap::swap_xy_i4 (integer(i4), intent(inout) *x*, integer(i4), intent(inout) *y*)

Definition at line 795 of file mo_utils.f90.

15.194.2.6 elemental pure subroutine mo_utils::swap::swap_xy_sp (real(sp), intent(inout) *x*, real(sp), intent(inout) *y*)

Definition at line 780 of file mo_utils.f90.

The documentation for this interface was generated from the following file:

- mo_utils.f90

15.195 mo_global_variables::twsstructure Type Reference

Collaboration diagram for mo_global_variables::twsstructure:

Public Attributes

- integer(i4), dimension(:), allocatable basinid
- character(256), dimension(:), allocatable fname
- real(dp), dimension(:,:), allocatable tws

15.195.1 Detailed Description

Definition at line 92 of file mo_global_variables.f90.

15.195.2 Member Data Documentation

15.195.2.1 integer(i4), dimension(:), allocatable mo_global_variables::twsstructure::basinid

Definition at line 93 of file mo_global_variables.f90.

15.195.2.2 character(256), dimension(:), allocatable mo_global_variables::twsstructure::fname

Definition at line 94 of file mo_global_variables.f90.

15.195.2.3 real(dp), dimension(:, :), allocatable mo_global_variables::twsstructure::tws

Definition at line 95 of file mo_global_variables.f90.

The documentation for this type was generated from the following file:

- mo_global_variables.f90

15.196 mo_orderpack::uniinv Interface Reference

Public Member Functions

- subroutine d_uniinv (XDONT, IGOEST)
- subroutine r_uniinv (XDONT, IGOEST)
- subroutine i_uniinv (XDONT, IGOEST)

15.196.1 Detailed Description

Definition at line 336 of file mo_orderpack.f90.

15.196.2 Member Function/Subroutine Documentation

15.196.2.1 subroutine mo_orderpack::uniinv::d_uniinv (real(kind = dp), dimension (:), intent(in) *XDONT,* integer(kind = i4), dimension (:), intent(out) *IGOEST*)

Definition at line 10230 of file mo_orderpack.f90.

15.196.2.2 subroutine mo_orderpack::uniinv::i_uniinv (integer(kind = i4), dimension (:), intent(in) *XDONT,* integer(kind = i4), dimension (:), intent(out) *IGOEST*)

Definition at line 10724 of file mo_orderpack.f90.

15.196.2.3 subroutine mo_orderpack::uniinv::r_uniinv (real(kind = sp), dimension (:), intent(in) *XDONT,* integer(kind = i4), dimension (:), intent(out) *IGOEST*)

Definition at line 10477 of file mo_orderpack.f90.

The documentation for this interface was generated from the following file:

- mo_orderpack.f90

15.197 mo_orderpack::unipar Interface Reference

Public Member Functions

- subroutine d_unipar (XDONT, IRNGT, NORD)
- subroutine r_unipar (XDONT, IRNGT, NORD)
- subroutine i_unipar (XDONT, IRNGT, NORD)

15.197.1 Detailed Description

Definition at line 342 of file mo_orderpack.f90.

15.197.2 Member Function/Subroutine Documentation

15.197.2.1 subroutine mo_orderpack::unipar::d_unipar (real(kind = dp), dimension (:), intent(in) *XDONT,* integer(kind = i4), dimension (:), intent(out) *IRNGT,* integer(kind = i4), intent(inout) *NORD*)

Definition at line 11005 of file mo_orderpack.f90.

15.197.2.2 subroutine mo_orderpack::unipar::i_unipar (integer(kind = i4), dimension (:), intent(in) *XDONT,* integer(kind = i4), dimension (:), intent(out) *IRNGT,* integer(kind = i4), intent(inout) *NORD*)

Definition at line 12207 of file mo_orderpack.f90.

15.197.2.3 subroutine mo_orderpack::unipar::r_unipar (real(kind = sp), dimension (:), intent(in) *XDONT,* integer(kind = i4), dimension (:), intent(out) *IRNGT,* integer(kind = i4), intent(inout) *NORD*)

Definition at line 11606 of file mo_orderpack.f90.

The documentation for this interface was generated from the following file:

- mo_orderpack.f90

15.198 mo_orderpack::unirnk Interface Reference

Public Member Functions

- subroutine d_unirnk (XVALT, IRNGT, NUNI)
- subroutine r_unirnk (XVALT, IRNGT, NUNI)
- subroutine i_unirnk (XVALT, IRNGT, NUNI)

15.198.1 Detailed Description

Definition at line 345 of file mo_orderpack.f90.

15.198.2 Member Function/Subroutine Documentation

15.198.2.1 subroutine mo_orderpack::unirnk::d_unirnk (real(kind = dp), dimension (:), intent(in) *XVALT,* integer(kind = i4), dimension (:), intent(out) *IRNGT,* integer(kind = i4), intent(out) *NUNI*)

Definition at line 12808 of file mo_orderpack.f90.

15.198.2.2 subroutine mo_orderpack::unirnk::i_unirnk (integer(kind = i4), dimension (:), intent(in) *XVALT,* integer(kind = i4), dimension (:), intent(out) *IRNGT,* integer(kind = i4), intent(out) *NUNI*)

Definition at line 13304 of file mo_orderpack.f90.

15.198.2.3 subroutine mo_orderpack::unirnk::r_unirnk (real(kind = sp), dimension (:), intent(in) *XVALT,* integer(kind = i4), dimension (:), intent(out) *IRNGT,* integer(kind = i4), intent(out) *NUNI*)

Definition at line 13056 of file mo_orderpack.f90.

The documentation for this interface was generated from the following file:

- mo_orderpack.f90

15.199 mo_orderpack::unista Interface Reference

Public Member Functions

- subroutine d_unista (XDONT, NUNI)

- subroutine r_unista (XDONT, NUNI)
- subroutine i_unista (XDONT, NUNI)

15.199.1 Detailed Description

Definition at line 348 of file mo_orderpack.f90.

15.199.2 Member Function/Subroutine Documentation

15.199.2.1 subroutine mo_orderpack::unista::d_unista (real(kind = dp), dimension (:), intent(inout) *XDONT,* integer(kind = i4), intent(out) *NUNI*)

Definition at line 13553 of file mo_orderpack.f90.

15.199.2.2 subroutine mo_orderpack::unista::i_unista (integer(kind = i4), dimension (:), intent(inout) *XDONT,* integer(kind = i4), intent(out) *NUNI*)

Definition at line 13611 of file mo_orderpack.f90.

15.199.2.3 subroutine mo_orderpack::unista::r_unista (real(kind = sp), dimension (:), intent(inout) *XDONT,* integer(kind = i4), intent(out) *NUNI*)

Definition at line 13582 of file mo_orderpack.f90.

The documentation for this interface was generated from the following file:

- mo_orderpack.f90

15.200 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)

15.200.1 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(Nc↩ Dimension), 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_restart.f90.

15.200.2 Member Function/Subroutine Documentation

15.200.2.1 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 788 of file mo_restart.f90.

15.200.2.2 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 743 of file mo_restart.f90.

15.200.2.3 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 833 of file mo_restart.f90.

15.200.2.4 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 890 of file mo_restart.f90.

The documentation for this interface was generated from the following file:

- mo_restart.f90

15.201 mo_orderpack::valmed Interface Reference

Public Member Functions

- recursive real(kind=dp) function d_valmed (XDONT)
- recursive real(kind=sp) function r_valmed (XDONT)
- recursive integer(kind=i4) function i_valmed (XDONT)

15.201.1 Detailed Description

Definition at line 351 of file mo_orderpack.f90.

15.201.2 Member Function/Subroutine Documentation

15.201.2.1 recursive real(kind = dp) function mo_orderpack::valmed::d_valmed (real(kind = dp), dimension (:), intent(in) *XDONT*)

Definition at line 13640 of file mo_orderpack.f90.

Referenced by mo_orderpack::d_valmed().

Here is the caller graph for this function:

15.201.2.2 recursive integer(kind = i4) function mo_orderpack::valmed::i_valmed (integer(kind = i4), dimension (:), intent(in) *XDONT*)

Definition at line 14312 of file mo_orderpack.f90.

Referenced by mo_orderpack::i_valmed().

Here is the caller graph for this function:

15.201.2.3 recursive real(kind = sp) function mo_orderpack::valmed::r_valmed (real(kind = sp), dimension (:), intent(in) *XDONT*)

Definition at line 13976 of file mo_orderpack.f90.

Referenced by mo_orderpack::r_valmed().

Here is the caller graph for this function:

The documentation for this interface was generated from the following file:

- mo_orderpack.f90

15.202 mo_orderpack::valnth Interface Reference

Public Member Functions

- real(kind=dp) function d_valnth (XDONT, NORD)
- real(kind=sp) function r_valnth (XDONT, NORD)
- integer(kind=i4) function i_valnth (XDONT, NORD)

15.202.1 Detailed Description

Definition at line 354 of file mo_orderpack.f90.

15.202.2 Member Function/Subroutine Documentation

15.202.2.1 real(kind = dp) function mo_orderpack::valnth::d_valnth (real(kind = dp), dimension (:), intent(in) *XDONT*, integer(kind = i4), intent(in) *NORD*)

Definition at line 14648 of file mo_orderpack.f90.

15.202.2.2 integer(kind = i4) function mo_orderpack::valnth::i_valnth (integer(kind = i4), dimension (:), intent(in) *XDONT*, integer(kind = i4), intent(in) *NORD*)

Definition at line 15634 of file mo_orderpack.f90.

15.202.2.3 real(kind = sp) function mo_orderpack::valnth::r_valnth (real(kind = sp), dimension (:), intent(in) *XDONT*, integer(kind = i4), intent(in) *NORD*)

Definition at line 15141 of file mo_orderpack.f90.

The documentation for this interface was generated from the following file:

- mo_orderpack.f90

15.203 mo_ncwrite::var2nc Interface Reference

Extended dump_netcdf for multiple variables.

Public Member Functions

- subroutine var2nc_1d_i4 (f_name, arr, dnames, v_name, dim_unlimited, long_name, units, missing_value, attributes, create, ncid, nrec)
- subroutine var2nc_1d_sp (f_name, arr, dnames, v_name, dim_unlimited, long_name, units, missing_value, attributes, create, ncid, nrec)
- subroutine var2nc_1d_dp (f_name, arr, dnames, v_name, dim_unlimited, long_name, units, missing_value, attributes, create, ncid, nrec)
- subroutine var2nc_2d_i4 (f_name, arr, dnames, v_name, dim_unlimited, long_name, units, missing_value, attributes, create, ncid, nrec)
- subroutine var2nc_2d_sp (f_name, arr, dnames, v_name, dim_unlimited, long_name, units, missing_value, attributes, create, ncid, nrec)
- subroutine var2nc_2d_dp (f_name, arr, dnames, v_name, dim_unlimited, long_name, units, missing_value, attributes, create, ncid, nrec)
- subroutine var2nc_3d_i4 (f_name, arr, dnames, v_name, dim_unlimited, long_name, units, missing_value, attributes, create, ncid, nrec)
- subroutine var2nc_3d_sp (f_name, arr, dnames, v_name, dim_unlimited, long_name, units, missing_value, attributes, create, ncid, nrec)
- subroutine var2nc_3d_dp (f_name, arr, dnames, v_name, dim_unlimited, long_name, units, missing_value, attributes, create, ncid, nrec)
- subroutine var2nc_4d_i4 (f_name, arr, dnames, v_name, dim_unlimited, long_name, units, missing_value, attributes, create, ncid, nrec)
- subroutine var2nc_4d_sp (f_name, arr, dnames, v_name, dim_unlimited, long_name, units, missing_value, attributes, create, ncid, nrec)
- subroutine var2nc_4d_dp (f_name, arr, dnames, v_name, dim_unlimited, long_name, units, missing_value, attributes, create, ncid, nrec)
- subroutine var2nc_5d_i4 (f_name, arr, dnames, v_name, dim_unlimited, long_name, units, missing_value, attributes, create, ncid, nrec)
- subroutine var2nc_5d_sp (f_name, arr, dnames, v_name, dim_unlimited, long_name, units, missing_value, attributes, create, ncid, nrec)
- subroutine var2nc_5d_dp (f_name, arr, dnames, v_name, dim_unlimited, long_name, units, missing_value, attributes, create, ncid, nrec)

15.203.1 Detailed Description

Extended dump_netcdf for multiple variables.

Write different variables including attributes to netcdf file. The attributes are restricted to long_name, units, and missing_value. It is also possible to append variables when an unlimited dimension is specified. call var2nc(f_↩name, arr, dnames, v_name, dim_unlimited, long_name, units, missing_value, create)

Parameters

in	*character(∗) :: f_name*	filename
in	*integer(i4)/real(sp,dp) :: arr(:[,:[,:[,:[,:]]]])*	array to write
in	*character(∗) :: dnames(:)*	dimension names
in	*character(∗) :: v_name*	variable name
in	*integer(i4), optional :: dim_unlimited*	index of unlimited dimension
in	*character(∗), optional :: long_name*	attribute
in	*character(∗), optional :: units*	attribute
in	*integer(i4)/real(sp,dp), optional :: missing_value*	attribute
in	*character(256), dimension(:,:), optional :: attributes*	two dimensional array of attributes size of first dimension equals number of attributes first entry of second dimension equals attribute name (e.g. long_↩name) second entry of second dimension equals attribute value (e.g. precipitation) every attribute is written as string with the exception of missing_value
in	*logical, optional :: create*	flag - specify whether a output file should be created, default
in,out	*integer(i4)/real(sp,dp), optional :: ncid*	if not given filename will be opened and closed if given and <0 then file will be opened and ncid will return the file unit. if given and >0 then file is assumed open and ncid is used as file unit.
in	*integer(i4), optional :: nrec*	if given: start point on unlimited dimension.

Note

It is not allowed to write the folloing numbers for the indicated type
number | kind
-2.1474836E+09 | integer(i4)
9.9692100E+36 | real(sp)
9.9692099683868690E+36 | real(dp)
These numbers are netcdf fortran 90 constants! They are used to determine the chunksize of the already written variable. Hence, this routine cannot append correctly to variables when these numbers are used. Only five dimensional variables can be written, only one unlimited dimension can be defined.

Author

Stephan Thober & Matthias Cuntz

Date

May 2014

Definition at line 301 of file mo_ncwrite.f90.

15.203.2 Member Function/Subroutine Documentation

15.203.2.1 **subroutine mo_ncwrite::var2nc::var2nc_1d_dp (character(len = ∗), intent(in) *f_name,* real(dp), dimension(:), intent(in) *arr,* character(len = ∗), dimension(:), intent(in) *dnames,* character(len = ∗), intent(in) *v_name,* integer(i4), intent(in), optional *dim_unlimited,* character(len = ∗), intent(in), optional *long_name,* character(len = ∗), intent(in), optional *units,* real(dp), intent(in), optional *missing_value,* character(256), dimension(:, :), intent(in), optional *attributes,* logical, intent(in), optional *create,* integer(i4), intent(inout), optional *ncid,* integer(i4), intent(in), optional *nrec*)**

Definition at line 3100 of file mo_ncwrite.f90.

15.203.2.2 **subroutine mo_ncwrite::var2nc::var2nc_1d_i4 (character(len = ∗), intent(in) *f_name,* integer(i4), dimension(:), intent(in) *arr,* character(len = ∗), dimension(:), intent(in) *dnames,* character(len = ∗), intent(in) *v_name,* integer(i4), intent(in), optional *dim_unlimited,* character(len = ∗), intent(in), optional *long_name,* character(len = ∗), intent(in), optional *units,* integer(i4), intent(in), optional *missing_value,* character(256), dimension(:, :), intent(in), optional *attributes,* logical, intent(in), optional *create,* integer(i4), intent(inout), optional *ncid,* integer(i4), intent(in), optional *nrec*)**

Definition at line 2759 of file mo_ncwrite.f90.

15.203.2.3 **subroutine mo_ncwrite::var2nc::var2nc_1d_sp (character(len = ∗), intent(in) *f_name,* real(sp), dimension(:), intent(in) *arr,* character(len = ∗), dimension(:), intent(in) *dnames,* character(len = ∗), intent(in) *v_name,* integer(i4), intent(in), optional *dim_unlimited,* character(len = ∗), intent(in), optional *long_name,* character(len = ∗), intent(in), optional *units,* real(sp), intent(in), optional *missing_value,* character(256), dimension(:, :), intent(in), optional *attributes,* logical, intent(in), optional *create,* integer(i4), intent(inout), optional *ncid,* integer(i4), intent(in), optional *nrec*)**

Definition at line 2930 of file mo_ncwrite.f90.

15.203.2.4 **subroutine mo_ncwrite::var2nc::var2nc_2d_dp (character(len = ∗), intent(in) *f_name,* real(dp), dimension(:, :), intent(in) *arr,* character(len = ∗), dimension(:), intent(in) *dnames,* character(len = ∗), intent(in) *v_name,* integer(i4), intent(in), optional *dim_unlimited,* character(len = ∗), intent(in), optional *long_name,* character(len = ∗), intent(in), optional *units,* real(dp), intent(in), optional *missing_value,* character(256), dimension(:, :), intent(in), optional *attributes,* logical, intent(in), optional *create,* integer(i4), intent(inout), optional *ncid,* integer(i4), intent(in), optional *nrec*)**

Definition at line 3630 of file mo_ncwrite.f90.

15.203.2.5 **subroutine mo_ncwrite::var2nc::var2nc_2d_i4 (character(len = ∗), intent(in) *f_name,* integer(i4), dimension(:, :), intent(in) *arr,* character(len = ∗), dimension(:), intent(in) *dnames,* character(len = ∗), intent(in) *v_name,* integer(i4), intent(in), optional *dim_unlimited,* character(len = ∗), intent(in), optional *long_name,* character(len = ∗), intent(in), optional *units,* integer(i4), intent(in), optional *missing_value,* character(256), dimension(:, :), intent(in), optional *attributes,* logical, intent(in), optional *create,* integer(i4), intent(inout), optional *ncid,* integer(i4), intent(in), optional *nrec*)**

Definition at line 3270 of file mo_ncwrite.f90.

15.203.2.6 subroutine mo_ncwrite::var2nc::var2nc_2d_sp (character(len = ∗), intent(in) *f_name,* real(sp), dimension(:, :), intent(in) *arr,* character(len = ∗), dimension(:), intent(in) *dnames,* character(len = ∗), intent(in) *v_name,* integer(i4), intent(in), optional *dim_unlimited,* character(len = ∗), intent(in), optional *long_name,* character(len = ∗), intent(in), optional *units,* real(sp), intent(in), optional *missing_value,* character(256), dimension(:, :), intent(in), optional *attributes,* logical, intent(in), optional *create,* integer(i4), intent(inout), optional *ncid,* integer(i4), intent(in), optional *nrec*)

Definition at line 3450 of file mo_ncwrite.f90.

15.203.2.7 subroutine mo_ncwrite::var2nc::var2nc_3d_dp (character(len = ∗), intent(in) *f_name,* real(dp), dimension(:, :, :), intent(in) *arr,* character(len = ∗), dimension(:), intent(in) *dnames,* character(len = ∗), intent(in) *v_name,* integer(i4), intent(in), optional *dim_unlimited,* character(len = ∗), intent(in), optional *long_name,* character(len = ∗), intent(in), optional *units,* real(dp), intent(in), optional *missing_value,* character(256), dimension(:, :), intent(in), optional *attributes,* logical, intent(in), optional *create,* integer(i4), intent(inout), optional *ncid,* integer(i4), intent(in), optional *nrec*)

Definition at line 4171 of file mo_ncwrite.f90.

15.203.2.8 subroutine mo_ncwrite::var2nc::var2nc_3d_i4 (character(len = ∗), intent(in) *f_name,* integer(i4), dimension(:, :, :), intent(in) *arr,* character(len = ∗), dimension(:), intent(in) *dnames,* character(len = ∗), intent(in) *v_name,* integer(i4), intent(in), optional *dim_unlimited,* character(len = ∗), intent(in), optional *long_name,* character(len = ∗), intent(in), optional *units,* integer(i4), intent(in), optional *missing_value,* character(256), dimension(:, :), intent(in), optional *attributes,* logical, intent(in), optional *create,* integer(i4), intent(inout), optional *ncid,* integer(i4), intent(in), optional *nrec*)

Definition at line 3810 of file mo_ncwrite.f90.

15.203.2.9 subroutine mo_ncwrite::var2nc::var2nc_3d_sp (character(len = ∗), intent(in) *f_name,* real(sp), dimension(:, :, :), intent(in) *arr,* character(len = ∗), dimension(:), intent(in) *dnames,* character(len = ∗), intent(in) *v_name,* integer(i4), intent(in), optional *dim_unlimited,* character(len = ∗), intent(in), optional *long_name,* character(len = ∗), intent(in), optional *units,* real(sp), intent(in), optional *missing_value,* character(256), dimension(:, :), intent(in), optional *attributes,* logical, intent(in), optional *create,* integer(i4), intent(inout), optional *ncid,* integer(i4), intent(in), optional *nrec*)

Definition at line 3990 of file mo_ncwrite.f90.

15.203.2.10 subroutine mo_ncwrite::var2nc::var2nc_4d_dp (character(len = ∗), intent(in) *f_name,* real(dp), dimension(:, :, :, :), intent(in) *arr,* character(len = ∗), dimension(:), intent(in) *dnames,* character(len = ∗), intent(in) *v_name,* integer(i4), intent(in), optional *dim_unlimited,* character(len = ∗), intent(in), optional *long_name,* character(len = ∗), intent(in), optional *units,* real(dp), intent(in), optional *missing_value,* character(256), dimension(:, :), intent(in), optional *attributes,* logical, intent(in), optional *create,* integer(i4), intent(inout), optional *ncid,* integer(i4), intent(in), optional *nrec*)

Definition at line 4715 of file mo_ncwrite.f90.

15.203.2.11 subroutine mo_ncwrite::var2nc::var2nc_4d_i4 (character(len = ∗), intent(in) *f_name,* integer(i4), dimension(:, :, :, :), intent(in) *arr,* character(len = ∗), dimension(:), intent(in) *dnames,* character(len = ∗), intent(in) *v_name,* integer(i4), intent(in), optional *dim_unlimited,* character(len = ∗), intent(in), optional *long_name,* character(len = ∗), intent(in), optional *units,* integer(i4), intent(in), optional *missing_value,* character(256), dimension(:, :), intent(in), optional *attributes,* logical, intent(in), optional *create,* integer(i4), intent(inout), optional *ncid,* integer(i4), intent(in), optional *nrec*)

Definition at line 4351 of file mo_ncwrite.f90.

15.203.2.12 subroutine mo_ncwrite::var2nc::var2nc_4d_sp (character(len = ∗), intent(in) *f_name,* real(sp), dimension(:, :, :, :), intent(in) *arr,* character(len = ∗), dimension(:), intent(in) *dnames,* character(len = ∗), intent(in) *v_name,* integer(i4), intent(in), optional *dim_unlimited,* character(len = ∗), intent(in), optional *long_name,* character(len = ∗), intent(in), optional *units,* real(sp), intent(in), optional *missing_value,* character(256), dimension(:, :), intent(in), optional *attributes,* logical, intent(in), optional *create,* integer(i4), intent(inout), optional *ncid,* integer(i4), intent(in), optional *nrec*)

Definition at line 4533 of file mo_ncwrite.f90.

15.203.2.13 subroutine mo_ncwrite::var2nc::var2nc_5d_dp (character(len = ∗), intent(in) *f_name,* real(dp), dimension(:, :, :, :, :), intent(in) *arr,* character(len = ∗), dimension(:), intent(in) *dnames,* character(len = ∗), intent(in) *v_name,* integer(i4), intent(in), optional *dim_unlimited,* character(len = ∗), intent(in), optional *long_name,* character(len = ∗), intent(in), optional *units,* real(dp), intent(in), optional *missing_value,* character(256), dimension(:, :), intent(in), optional *attributes,* logical, intent(in), optional *create,* integer(i4), intent(inout), optional *ncid,* integer(i4), intent(in), optional *nrec*)

Definition at line 5241 of file mo_ncwrite.f90.

15.203.2.14 subroutine mo_ncwrite::var2nc::var2nc_5d_i4 (character(len = ∗), intent(in) *f_name,* integer(i4), dimension(:, :, :, :, :), intent(in) *arr,* character(len = ∗), dimension(:), intent(in) *dnames,* character(len = ∗), intent(in) *v_name,* integer(i4), intent(in), optional *dim_unlimited,* character(len = ∗), intent(in), optional *long_name,* character(len = ∗), intent(in), optional *units,* integer(i4), intent(in), optional *missing_value,* character(256), dimension(:, :), intent(in), optional *attributes,* logical, intent(in), optional *create,* integer(i4), intent(inout), optional *ncid,* integer(i4), intent(in), optional *nrec*)

Definition at line 4897 of file mo_ncwrite.f90.

15.203.2.15 subroutine mo_ncwrite::var2nc::var2nc_5d_sp (character(len = ∗), intent(in) *f_name,* real(sp), dimension(:, :, :, :, :), intent(in) *arr,* character(len = ∗), dimension(:), intent(in) *dnames,* character(len = ∗), intent(in) *v_name,* integer(i4), intent(in), optional *dim_unlimited,* character(len = ∗), intent(in), optional *long_name,* character(len = ∗), intent(in), optional *units,* real(sp), intent(in), optional *missing_value,* character(256), dimension(:, :), intent(in), optional *attributes,* logical, intent(in), optional *create,* integer(i4), intent(inout), optional *ncid,* integer(i4), intent(in), optional *nrec*)

Definition at line 5069 of file mo_ncwrite.f90.

The documentation for this interface was generated from the following file:

- mo_ncwrite.f90

15.204 mo_ncwrite::variable Type Reference

Collaboration diagram for mo_ncwrite::variable:

Public Attributes

- character(len=maxlen) name
- integer(i4) xtype
- integer(i4) nlvls

- integer(i4) nsubs
- logical unlimited
- integer(i4) varid
- integer(i4) ndims
- integer(i4), dimension(nmaxdim) dimids
- integer(i4), dimension(nmaxdim) dimtypes
- integer(i4) natt
- type(attribute), dimension(nmaxatt) att
- integer(i4), dimension(nmaxdim) start
- integer(i4), dimension(nmaxdim) count
- logical wflag
- integer(i1), pointer g0_b
- integer(i1), dimension(:), pointer g1_b
- integer(i1), dimension(:,:), pointer g2_b
- integer(i1), dimension(:,:,:), pointer g3_b
- integer(i1), dimension(:,:,:,:), pointer g4_b
- integer(i4), pointer g0_i
- integer(i4), dimension(:), pointer g1_i
- integer(i4), dimension(:,:), pointer g2_i
- integer(i4), dimension(:,:,:), pointer g3_i
- integer(i4), dimension(:,:,:,:), pointer g4_i
- real(sp), pointer g0_f
- real(sp), dimension(:), pointer g1_f
- real(sp), dimension(:,:), pointer g2_f
- real(sp), dimension(:,:,:), pointer g3_f
- real(sp), dimension(:,:,:,:), pointer g4_f
- real(dp), pointer g0_d
- real(dp), dimension(:), pointer g1_d
- real(dp), dimension(:,:), pointer g2_d
- real(dp), dimension(:,:,:), pointer g3_d
- real(dp), dimension(:,:,:,:), pointer g4_d

15.204.1 Detailed Description

Definition at line 72 of file mo_ncwrite.f90.

15.204.2 Member Data Documentation

15.204.2.1 type(attribute), dimension(nmaxatt) mo_ncwrite::variable::att

Definition at line 83 of file mo_ncwrite.f90.

15.204.2.2 integer(i4), dimension(nmaxdim) mo_ncwrite::variable::count

Definition at line 85 of file mo_ncwrite.f90.

15.204.2.3 integer(i4), dimension(nmaxdim) mo_ncwrite::variable::dimids

Definition at line 80 of file mo_ncwrite.f90.

15.204.2.4 integer(i4), dimension(nmaxdim) mo_ncwrite::variable::dimtypes

Definition at line 81 of file mo_ncwrite.f90.

15.204.2.5 integer(i1), pointer mo_ncwrite::variable::g0_b

Definition at line 87 of file mo_ncwrite.f90.

15.204.2.6 real(dp), pointer mo_ncwrite::variable::g0_d

Definition at line 102 of file mo_ncwrite.f90.

15.204.2.7 real(sp), pointer mo_ncwrite::variable::g0_f

Definition at line 97 of file mo_ncwrite.f90.

15.204.2.8 integer(i4), pointer mo_ncwrite::variable::g0_i

Definition at line 92 of file mo_ncwrite.f90.

15.204.2.9 integer(i1), dimension(:), pointer mo_ncwrite::variable::g1_b

Definition at line 88 of file mo_ncwrite.f90.

15.204.2.10 real(dp), dimension(:), pointer mo_ncwrite::variable::g1_d

Definition at line 103 of file mo_ncwrite.f90.

15.204.2.11 real(sp), dimension(:), pointer mo_ncwrite::variable::g1_f

Definition at line 98 of file mo_ncwrite.f90.

15.204.2.12 integer(i4), dimension(:), pointer mo_ncwrite::variable::g1_i

Definition at line 93 of file mo_ncwrite.f90.

15.204.2.13 integer(i1), dimension(:, :), pointer mo_ncwrite::variable::g2_b

Definition at line 89 of file mo_ncwrite.f90.

15.204.2.14 real(dp), dimension(:, :), pointer mo_ncwrite::variable::g2_d

Definition at line 104 of file mo_ncwrite.f90.

15.204.2.15 real(sp), dimension(:, :), pointer mo_ncwrite::variable::g2_f

Definition at line 99 of file mo_ncwrite.f90.

15.204.2.16 integer(i4), dimension(:, :), pointer mo_ncwrite::variable::g2_i

Definition at line 94 of file mo_ncwrite.f90.

15.204.2.17 integer(i1), dimension(:, :, :), pointer mo_ncwrite::variable::g3_b

Definition at line 90 of file mo_ncwrite.f90.

15.204.2.18 real(dp), dimension(:, :, :), pointer mo_ncwrite::variable::g3_d

Definition at line 105 of file mo_ncwrite.f90.

15.204.2.19 real(sp), dimension(:, :, :), pointer mo_ncwrite::variable::g3_f

Definition at line 100 of file mo_ncwrite.f90.

15.204.2.20 integer(i4), dimension(:, :, :), pointer mo_ncwrite::variable::g3_i

Definition at line 95 of file mo_ncwrite.f90.

15.204.2.21 integer(i1), dimension(:, :, :, :), pointer mo_ncwrite::variable::g4_b

Definition at line 91 of file mo_ncwrite.f90.

15.204.2.22 real(dp), dimension(:, :, :, :), pointer mo_ncwrite::variable::g4_d

Definition at line 106 of file mo_ncwrite.f90.

15.204.2.23 real(sp), dimension(:, :, :, :), pointer mo_ncwrite::variable::g4_f

Definition at line 101 of file mo_ncwrite.f90.

15.204.2.24 integer(i4), dimension(:, :, :, :), pointer mo_ncwrite::variable::g4_i

Definition at line 96 of file mo_ncwrite.f90.

15.204.2.25 character (len = maxlen) mo_ncwrite::variable::name

Definition at line 73 of file mo_ncwrite.f90.

15.204.2.26 integer(i4) mo_ncwrite::variable::natt

Definition at line 82 of file mo_ncwrite.f90.

15.204.2.27 integer(i4) mo_ncwrite::variable::ndims

Definition at line 79 of file mo_ncwrite.f90.

15.204.2.28 integer(i4) mo_ncwrite::variable::nlvls

Definition at line 75 of file mo_ncwrite.f90.

15.204.2.29 integer(i4) mo_ncwrite::variable::nsubs

Definition at line 76 of file mo_ncwrite.f90.

15.204.2.30 integer(i4), dimension(nmaxdim) mo_ncwrite::variable::start

Definition at line 84 of file mo_ncwrite.f90.

15.204.2.31 logical mo_ncwrite::variable::unlimited

Definition at line 77 of file mo_ncwrite.f90.

15.204.2.32 integer(i4) mo_ncwrite::variable::varid

Definition at line 78 of file mo_ncwrite.f90.

15.204.2.33 logical mo_ncwrite::variable::wflag

Definition at line 86 of file mo_ncwrite.f90.

15.204.2.34 integer(i4) mo_ncwrite::variable::xtype

Definition at line 74 of file mo_ncwrite.f90.

The documentation for this type was generated from the following file:

- mo_ncwrite.f90

15.205 mo_moment::variance Interface Reference

Public Member Functions

- real(sp) function variance_sp (dat, mask)
- real(dp) function variance_dp (dat, mask)

15.205.1 Detailed Description

Definition at line 835 of file mo_moment.f90.

15.205.2 Member Function/Subroutine Documentation

15.205.2.1 real(dp) function mo_moment::variance::variance_dp (real(dp), dimension(:), intent(in) *dat,* logical, dimension(:), intent(in), optional *mask*)

Definition at line 1890 of file mo_moment.f90.

Referenced by mo_moment::variance_dp().

Here is the caller graph for this function:

15.205.2.2 real(sp) function mo_moment::variance::variance_sp (real(sp), dimension(:), intent(in) *dat,* logical, dimension(:), intent(in), optional *mask*)

Definition at line 1926 of file mo_moment.f90.

Referenced by mo_moment::variance_sp().

Here is the caller graph for this function:

The documentation for this interface was generated from the following file:

- mo_moment.f90

15.206 mo_errormeasures::wnse Interface Reference

Public Member Functions

- real(sp) function wnse_sp_1d (x, y, mask)
- real(dp) function wnse_dp_1d (x, y, mask)
- real(dp) function wnse_dp_2d (x, y, mask)
- real(sp) function wnse_sp_2d (x, y, mask)
- real(sp) function wnse_sp_3d (x, y, mask)
- real(dp) function wnse_dp_3d (x, y, mask)

15.206.1 Detailed Description

Definition at line 768 of file mo_errormeasures.f90.

15.206.2 Member Function/Subroutine Documentation

15.206.2.1 real(dp) function mo_errormeasures::wnse::wnse_dp_1d (real(dp), dimension(:), intent(in) *x*, real(dp), dimension(:), intent(in) *y*, logical, dimension(:), intent(in), optional *mask*)

Definition at line 3386 of file mo_errormeasures.f90.

Referenced by mo_errormeasures::wnse_dp_1d().

Here is the caller graph for this function:

15.206.2.2 real(dp) function mo_errormeasures::wnse::wnse_dp_2d (real(dp), dimension(:,:), intent(in) *x*, real(dp), dimension(:,:), intent(in) *y*, logical, dimension(:,:), intent(in), optional *mask*)

Definition at line 3469 of file mo_errormeasures.f90.

Referenced by mo_errormeasures::wnse_dp_2d().

Here is the caller graph for this function:

15.206.2.3 real(dp) function mo_errormeasures::wnse::wnse_dp_3d (real(dp), dimension(:,:,:), intent(in) *x*, real(dp), dimension(:,:,:), intent(in) *y*, logical, dimension(:,:,:), intent(in), optional *mask*)

Definition at line 3550 of file mo_errormeasures.f90.

Referenced by mo_errormeasures::wnse_dp_3d().

Here is the caller graph for this function:

15.206.2.4 real(sp) function mo_errormeasures::wnse::wnse_sp_1d (real(sp), dimension(:), intent(in) *x*, real(sp), dimension(:), intent(in) *y*, logical, dimension(:), intent(in), optional *mask*)

Definition at line 3343 of file mo_errormeasures.f90.

Referenced by mo_errormeasures::wnse_sp_1d().

Here is the caller graph for this function:

15.206.2.5 real(sp) function mo_errormeasures::wnse::wnse_sp_2d (real(sp), dimension(:,:), intent(in) *x*, real(sp), dimension(:,:), intent(in) *y*, logical, dimension(:,:), intent(in), optional *mask*)

Definition at line 3429 of file mo_errormeasures.f90.

Referenced by mo_errormeasures::wnse_sp_2d().

Here is the caller graph for this function:

15.206.2.6 real(sp) function mo_errormeasures::wnse::wnse_sp_3d (real(sp), dimension(:,:,:), intent(in) *x*, real(sp), dimension(:,:,:), intent(in) *y*, logical, dimension(:,:,:), intent(in), optional *mask*)

Definition at line 3509 of file mo_errormeasures.f90.

Referenced by mo_errormeasures::wnse_sp_3d().

Here is the caller graph for this function:

The documentation for this interface was generated from the following file:

- mo_errormeasures.f90

15.207 mo_xor4096::xor4096 Interface Reference

Public Member Functions

- subroutine xor4096s_0d (seed, SingleIntegerRN, save_state)
- subroutine xor4096s_1d (seed, SingleIntegerRN, save_state)
- subroutine xor4096f_0d (seed, SingleRealRN, save_state)
- subroutine xor4096f_1d (seed, SingleRealRN, save_state)
- subroutine xor4096l_0d (seed, DoubleIntegerRN, save_state)
- subroutine xor4096l_1d (seed, DoubleIntegerRN, save_state)
- subroutine xor4096d_0d (seed, DoubleRealRN, save_state)
- subroutine xor4096d_1d (seed, DoubleRealRN, save_state)

15.207.1 Detailed Description

Definition at line 232 of file mo_xor4096.f90.

15.207.2 Member Function/Subroutine Documentation

15.207.2.1 subroutine mo_xor4096::xor4096::xor4096d_0d (integer(i8), intent(in) *seed*, real(dp), intent(out) *DoubleRealRN*, integer(i8), dimension(n_save_state), intent(inout), optional *save_state*)

Definition at line 1134 of file mo_xor4096.f90.

15.207.2.2 subroutine mo_xor4096::xor4096::xor4096d_1d (integer(i8), dimension(:), intent(in) *seed*, real(dp), dimension(size(seed, 1)), intent(out) *DoubleRealRN*, integer(i8), dimension(size(seed, 1), n_save_state), intent(inout), optional *save_state*)

Definition at line 1237 of file mo_xor4096.f90.

15.207.2.3 subroutine mo_xor4096::xor4096::xor4096f_0d (integer(i4), intent(in) *seed,* real(sp), intent(out) *SingleRealRN,* integer(i4), dimension(n_save_state), intent(inout), optional *save_state*)

Definition at line 665 of file mo_xor4096.f90.

15.207.2.4 subroutine mo_xor4096::xor4096::xor4096f_1d (integer(i4), dimension(:), intent(in) *seed,* real(sp), dimension(size(seed)), intent(out) *SingleRealRN,* integer(i4), dimension(size(seed, 1), n_save_state), intent(inout), optional *save_state*)

Definition at line 769 of file mo_xor4096.f90.

15.207.2.5 subroutine mo_xor4096::xor4096::xor4096l_0d (integer(i8), intent(in) *seed,* integer(i8), intent(out) *DoubleIntegerRN,* integer(i8), dimension(n_save_state), intent(inout), optional *save_state*)

Definition at line 910 of file mo_xor4096.f90.

15.207.2.6 subroutine mo_xor4096::xor4096::xor4096l_1d (integer(i8), dimension(:), intent(in) *seed,* integer(i8), dimension(size(seed, 1)), intent(out) *DoubleIntegerRN,* integer(i8), dimension(size(seed, 1), n_save_state), intent(inout), optional *save_state*)

Definition at line 1003 of file mo_xor4096.f90.

15.207.2.7 subroutine mo_xor4096::xor4096::xor4096s_0d (integer(i4), intent(in) *seed,* integer(i4), intent(out) *SingleIntegerRN,* integer(i4), dimension(n_save_state), intent(inout), optional *save_state*)

Definition at line 441 of file mo_xor4096.f90.

15.207.2.8 subroutine mo_xor4096::xor4096::xor4096s_1d (integer(i4), dimension(:), intent(in) *seed,* integer(i4), dimension(size(seed, 1)), intent(out) *SingleIntegerRN,* integer(i4), dimension(size(seed, 1), n_save_state), intent(inout), optional *save_state*)

Definition at line 534 of file mo_xor4096.f90.

The documentation for this interface was generated from the following file:

- mo_xor4096.f90

15.208 mo_xor4096::xor4096g Interface Reference

Public Member Functions

- subroutine xor4096gf_0d (seed, SingleRealRN, save_state)
- subroutine xor4096gf_1d (seed, SingleRealRN, save_state)
- subroutine xor4096gd_0d (seed, DoubleRealRN, save_state)
- subroutine xor4096gd_1d (seed, DoubleRealRN, save_state)

15.208.1 Detailed Description

Definition at line 346 of file mo_xor4096.f90.

15.208.2 Member Function/Subroutine Documentation

15.208.2.1 subroutine mo_xor4096::xor4096g::xor4096gd_0d (integer(i8), intent(in) *seed,* real(dp), intent(out) *DoubleRealRN,* integer(i8), dimension(n_save_state), intent(inout), optional *save_state*)

Definition at line 1742 of file mo_xor4096.f90.

15.208.2.2 subroutine mo_xor4096::xor4096g::xor4096gd_1d (integer(i8), dimension(:), intent(in) *seed,* real(dp), dimension(size(seed)), intent(out) *DoubleRealRN,* integer(i8), dimension(size(seed), n_save_state), intent(inout), optional *save_state*)

Definition at line 1897 of file mo_xor4096.f90.

15.208.2.3 subroutine mo_xor4096::xor4096g::xor4096gf_0d (integer(i4), intent(in) *seed,* real(sp), intent(out) *SingleRealRN,* integer(i4), dimension(n_save_state), intent(inout), optional *save_state*)

Definition at line 1377 of file mo_xor4096.f90.

15.208.2.4 subroutine mo_xor4096::xor4096g::xor4096gf_1d (integer(i4), dimension(:), intent(in) *seed,* real(sp), dimension(size(seed)), intent(out) *SingleRealRN,* integer(i4), dimension(size(seed), n_save_state), intent(inout), optional *save_state*)

Definition at line 1533 of file mo_xor4096.f90.

The documentation for this interface was generated from the following file:

- mo_xor4096.f90

Chapter 16

File Documentation

16.1 1-main.dox File Reference

16.2 2-get_started.dox File Reference

16.3 3-data_preparation.dox File Reference

16.4 4-visualise_out.dox File Reference

16.5 5-calibration.dox File Reference

16.6 6-style_guide.dox File Reference

16.7 7-test_basin.dox File Reference

16.8 8-protocols_for_setup_new_mHM_basin.dox File Reference

16.9 9-mRM.md File Reference

16.10 DEPENDENCIES.md File Reference

16.11 mhm_driver.f90 File Reference

Functions/Subroutines

- program mhm_driver

  Distributed precipitation-runoff model mHM.

16.11.1 Function/Subroutine Documentation

16.11.1.1 program mhm_driver ( )

Distributed precipitation-runoff model mHM.

This is the main driver of mHM, which calls one instance of mHM for a multiple basins and a given period.

Figure 16.1: Typical mHM cell

Authors

Luis Samaniego & Rohini Kumar (UFZ)

Date

Jun 2018

Version

5.9

Copyright

(c)2005-2018, Helmholtz-Zentrum fuer Umweltforschung GmbH - UFZ. All rights reserved. This code is a property of: ---------- Helmholtz-Zentrum fuer Umweltforschung GmbH - UFZ Registered Office: Leipzig Registration Office: Amtsgericht Leipzig Trade Register: Nr. B 4703 Chairman of the Supervisory Board: MinDirig Wilfried Kraus Scientific Director: Prof. Dr. Georg Teutsch Administrative Director: Dr. Heike Grassmann ---------- NEITHER UFZ NOR THE DE↩ VELOPERS MAKES ANY WARRANTY, EXPRESS OR IMPLIED, OR ASSUMES ANY LIABILITY FOR T↩ HE USE OF THIS SOFTWARE. If software is modified to produce derivative works, such modified 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 78 of file mhm_driver.f90.

References mo_common_read_config::common_read_config(), mo_finish::finish(), mo_message::message(), mo_mhm_eval::mhm_eval(), mo_startup::mhm_initialize(), mo_mhm_read_config::mhm_read_config(), mo_mrm_init::mrm_init(), mo_mrm_write::mrm_write(), mo_objective_function::objective(), mo_meteo_forcings::prepare_meteo_forcings_data(), mo_read_optional_data::read_basin_avg_tws(), mo_read_optional_data::read_evapotranspiration(), mo_read_optional_data::read_neutrons(), mo_read_optional_data::read_soil_moisture(), mo_mrm_objective_function_runoff::single_objective_runoff(), mo_timer::timer_get(), mo_timer::timer_start(), mo_timer::timer_stop(), mo_timer::timers_init(), mo_write_ascii::write_configfile(), mo_write_ascii::write_optifile(), mo_write_ascii::write_optinamelist(), and mo_restart::write_restart_files().

Here is the call graph for this function:

16.12 mhm_papers.md File Reference

16.13 mo_anneal.f90 File Reference

Data Types

- module mo_anneal
- interface mo_anneal::anneal

  anneal
- interface mo_anneal::gettemperature

  GetTemperature.
- interface mo_anneal::generate_neighborhood_weight

16.14 mo_append.f90 File Reference

Data Types

- module mo_append

  Append values on existing arrays.
- interface mo_append::append

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

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

16.15 mo_canopy_interc.f90 File Reference

Data Types

- module mo_canopy_interc

  Canopy interception.

16.16 mo_common_constants.f90 File Reference

Data Types

- module mo_common_constants

  Provides constants commonly used by mHM, mRM and MPR.

16.17 mo_common_file.f90 File Reference

Data Types

- module mo_common_file

  Provides file names and units for mRM.

16.18 mo_common_functions.f90 File Reference

Data Types

- module mo_common_functions

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

16.19 mo_common_read_config.f90 File Reference

Data Types

- module mo_common_read_config

  Reading of main model configurations.

16.20 mo_common_read_data.f90 File Reference

Data Types

- module mo_common_read_data

  TODO: add description.

16.21 mo_common_restart.f90 File Reference

Data Types

- module mo_common_restart

  TODO: add description.

16.22 mo_common_variables.f90 File Reference

Data Types

- module mo_common_variables

  Provides structures needed by mHM, mRM and/or mpr.
- type mo_common_variables::period
- type mo_common_variables::grid
- type mo_common_variables::gridremapper

16.23 mo_constants.f90 File Reference

Data Types

- module mo_constants

  Provides computational, mathematical, physical, and file constants.

16.24 mo_corr.f90 File Reference

Data Types

- module mo_corr
- interface mo_corr::autocoeffk
- interface mo_corr::autocorr
- interface mo_corr::corr
- interface mo_corr::crosscoeffk
- interface mo_corr::crosscorr
- interface mo_corr::arth
- interface mo_corr::four1
- interface mo_corr::fourrow
- interface mo_corr::realft
- interface mo_corr::swap

16.25 mo_dds.f90 File Reference

Data Types

- module mo_dds

  Dynamically Dimensioned Search (DDS)

16.26 mo_errormeasures.f90 File Reference

Data Types

- module mo_errormeasures
- interface mo_errormeasures::bias
- interface mo_errormeasures::kge

  Kling-Gupta-Efficiency measure.
- interface mo_errormeasures::kgenocorr

  Kling-Gupta-Efficiency measure without correlation.
- interface mo_errormeasures::lnnse
- interface mo_errormeasures::mae
- interface mo_errormeasures::mse
- interface mo_errormeasures::nse
- interface mo_errormeasures::sae
- interface mo_errormeasures::sse
- interface mo_errormeasures::rmse
- interface mo_errormeasures::wnse

16.27 mo_file.f90 File Reference

Data Types

- module mo_file

  Provides file names and units for mHM.

16.28 mo_finish.f90 File Reference

Data Types

- module mo_finish

  Finish a program gracefully.

16.29 mo_global_variables.f90 File Reference

Data Types

- module mo_global_variables

  Global variables ONLY used in reading, writing and startup.
- type mo_global_variables::twsstructure

16.30 mo_grid.f90 File Reference

Data Types

- module mo_grid

  TODO: add description.

16.31 mo_init_states.f90 File Reference

Data Types

- module mo_init_states

  Initialization of all state variables of mHM.

16.32 mo_julian.f90 File Reference

Data Types

- module mo_julian

  Julian date conversion routines.
- interface mo_julian::setcalendar

16.33 mo_kind.f90 File Reference

Data Types

- module mo_kind

  Define number representations.
- type mo_kind::sprs2_sp

  Single Precision Numerical Recipes types for sparse arrays.
- type mo_kind::sprs2_dp

  Double Precision Numerical Recipes types for sparse arrays.

16.34 mo_linfit.f90 File Reference

Data Types

- module mo_linfit

  Fitting a straight line.
- interface mo_linfit::linfit

  Fits a straight line to input data by minimizing chi^2.

16.35 mo_mad.f90 File Reference

Data Types

- module mo_mad
- interface mo_mad::mad

16.36 mo_mcmc.f90 File Reference

Data Types

- module mo_mcmc

  This module is Monte Carlo Markov Chain sampling of a posterior parameter distribution.
- interface mo_mcmc::mcmc

  This module is Monte Carlo Markov Chain sampling of a posterior parameter distribution (measurement errors are either known or modeled).
- interface mo_mcmc::mcmc_stddev

  This module is Monte Carlo Markov Chain sampling of a posterior parameter distribution (measurement errors are neither known nor modeled).

16.37 mo_message.f90 File Reference

Data Types

- module mo_message

  Write out concatenated strings.

16.38 mo_meteo_forcings.f90 File Reference

Data Types

- module mo_meteo_forcings

  Prepare meteorological forcings data for mHM.

16.39 mo_mhm.f90 File Reference

Data Types

- module mo_mhm

Call all main processes of mHM.

16.40 mo_mhm_constants.f90 File Reference

Data Types

- module mo_mhm_constants

  Provides mHM specific constants.

16.41 mo_mhm_eval.f90 File Reference

Data Types

- module mo_mhm_eval

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

16.42 mo_mhm_read_config.f90 File Reference

Data Types

- module mo_mhm_read_config

  Reading of main model configurations.

16.43 mo_moment.f90 File Reference

Data Types

- module mo_moment
- interface mo_moment::absdev
- interface mo_moment::average
- interface mo_moment::central_moment
- interface mo_moment::central_moment_var
- interface mo_moment::correlation
- interface mo_moment::covariance
- interface mo_moment::kurtosis
- interface mo_moment::mean
- interface mo_moment::mixed_central_moment
- interface mo_moment::mixed_central_moment_var
- interface mo_moment::moment
- interface mo_moment::skewness
- interface mo_moment::stddev
- interface mo_moment::variance

16.44 mo_mrm_constants.f90 File Reference

Data Types

- module mo_mrm_constants

  Provides mRM specific constants.

16.45 mo_mrm_eval.f90 File Reference

Data Types

- module mo_mrm_eval

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

16.46 mo_mrm_file.f90 File Reference

Data Types

- module mo_mrm_file

  Provides file names and units for mRM.

16.47 mo_mrm_global_variables.f90 File Reference

Data Types

- module mo_mrm_global_variables

  Global variables for mRM only.
- type mo_mrm_global_variables::gaugingstation
- type mo_mrm_global_variables::basininfo_mrm

16.48 mo_mrm_init.f90 File Reference

Data Types

- module mo_mrm_init

  Wrapper for initializing Routing.

16.49 mo_mrm_mpr.f90 File Reference

Data Types

- module mo_mrm_mpr

  Perform Multiscale Parameter Regionalization on Routing Parameters.

16.50 mo_mrm_net_startup.f90 File Reference

Data Types

- module mo_mrm_net_startup

  Startup drainage network for mHM.

Functions/Subroutines

- recursive subroutine calculate_l11_flow_accumulation (fDir, fAcc, ii, jj, nrow, ncol)

16.50.1 Function/Subroutine Documentation

16.50.1.1 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 2095 of file mo_mrm_net_startup.f90.

Referenced by mo_mrm_net_startup::l11_flow_accumulation().

Here is the caller graph for this function:

16.51 mo_mrm_objective_function_runoff.f90 File Reference

Data Types

- module mo_mrm_objective_function_runoff

  Objective Functions for Optimization of mHM/mRM against runoff.

16.52 mo_mrm_read_config.f90 File Reference

Data Types

- module mo_mrm_read_config

  read mRM config

16.53 mo_mrm_read_data.f90 File Reference

Data Types

- module mo_mrm_read_data

  This module contains all routines to read mRM data from file.

16.54 mo_mrm_restart.f90 File Reference

Data Types

- module mo_mrm_restart

  Restart routines.

16.55 mo_mrm_river_head.f90 File Reference

Data Types

- module mo_mrm_river_head

16.56 mo_mrm_routing.f90 File Reference

Data Types

- module mo_mrm_routing

  Performs runoff routing for mHM at level L11.

16.57 mo_mrm_signatures.f90 File Reference

Data Types

- module mo_mrm_signatures

  Module with calculations for several hydrological signatures.

16.58 mo_mrm_write.f90 File Reference

Data Types

- module mo_mrm_write

  write of discharge and restart files

16.59 mo_mrm_write_fluxes_states.f90 File Reference

Data Types

- module mo_mrm_write_fluxes_states

  Creates NetCDF output for different fluxes and state variables of mHM.
- interface mo_mrm_write_fluxes_states::outputvariable
- interface mo_mrm_write_fluxes_states::outputvariable
- interface mo_mrm_write_fluxes_states::outputdataset
- interface mo_mrm_write_fluxes_states::outputdataset

16.60 mo_ncread.f90 File Reference

Data Types

- module mo_ncread
- interface mo_ncread::get_ncvar

16.61 mo_ncwrite.f90 File Reference

Data Types

- module mo_ncwrite
- type mo_ncwrite::dims
- type mo_ncwrite::attribute
- type mo_ncwrite::variable
- interface mo_ncwrite::dump_netcdf
- interface mo_ncwrite::var2nc

  Extended dump_netcdf for multiple variables.

16.62 mo_netcdf.f90 File Reference

Data Types

- module mo_netcdf

  NetCDF Fortran 90 interface wrapper.
- interface mo_netcdf::ncdataset

  Provides basic file modification functionality.
- interface mo_netcdf::ncdataset

  Provides basic file modification functionality.
- type mo_netcdf::ncdimension

  Provides the dimension access functionality.
- interface mo_netcdf::ncvariable

16.63 mo_neutrons.f90 File Reference

Data Types

- module mo_neutrons

  Models to predict neutron intensities above soils.

16.64 mo_nml.f90 File Reference

Data Types

- module mo_nml

  Deal with namelist files.

16.65 mo_objective_function.f90 File Reference

Data Types

- module mo_objective_function

  Objective Functions for Optimization of mHM.

16.66 mo_optimization_utils.f90 File Reference

Data Types

- module mo_optimization_utils
- interface mo_optimization_utils::eval_interface
- interface mo_optimization_utils::objective_interface

16.67 mo_orderpack.f90 File Reference

Data Types

- module mo_orderpack

  Sort and ranking routines.
- interface mo_orderpack::sort

  Unconditional ranking.
- interface mo_orderpack::sort_index
- interface mo_orderpack::ctrper
- interface mo_orderpack::fndnth
- interface mo_orderpack::indmed
- interface mo_orderpack::indnth
- interface mo_orderpack::inspar
- interface mo_orderpack::inssor
- interface mo_orderpack::omedian
- interface mo_orderpack::mrgref
- interface mo_orderpack::mrgrnk
- interface mo_orderpack::mulcnt
- interface mo_orderpack::rapknr
- interface mo_orderpack::refpar
- interface mo_orderpack::refsor
- interface mo_orderpack::rinpar
- interface mo_orderpack::rnkpar
- interface mo_orderpack::uniinv
- interface mo_orderpack::nearless
- interface mo_orderpack::unipar
- interface mo_orderpack::unirnk
- interface mo_orderpack::unista
- interface mo_orderpack::valmed
- interface mo_orderpack::valnth

16.68 mo_percentile.f90 File Reference

Data Types

- module mo_percentile
- interface mo_percentile::median
- interface mo_percentile::n_element
- interface mo_percentile::percentile
- interface mo_percentile::qmedian

16.69 mo_pet.f90 File Reference

Data Types

- module mo_pet

  Module for calculating reference/potential evapotranspiration [mm s-1].

16.70 mo_read_forcing_nc.f90 File Reference

Data Types

- module mo_read_forcing_nc

  Reads forcing input data.

16.71 mo_read_latlon.f90 File Reference

Data Types

- module mo_read_latlon

  reading latitude and longitude coordinates for each basin

16.72 mo_read_optional_data.f90 File Reference

Data Types

- module mo_read_optional_data

  Read optional data for mHM calibration.

16.73 mo_read_spatial_data.f90 File Reference

Data Types

- module mo_read_spatial_data

  Reads spatial input data.
- interface mo_read_spatial_data::read_spatial_data_ascii

  Reads spatial data files of ASCII format.

16.74 mo_read_timeseries.f90 File Reference

Data Types

- module mo_read_timeseries

  Routines to read files containing timeseries data.

16.75 mo_restart.f90 File Reference

Data Types

- module mo_restart

  reading and writing states, fluxes and configuration for restart of mHM.
- interface mo_restart::unpack_field_and_write

  TODO: add description.

16.76 mo_runoff.f90 File Reference

Data Types

- module mo_runoff

  Runoff generation for the unsaturated zone, saturated zone (or groundwater zone), and runoff accumulation.

16.77 mo_sce.f90 File Reference

Data Types

- module mo_sce

  Shuffled Complex Evolution optimization algorithm.

Functions/Subroutines

- subroutine write_best_intermediate (to_file)
- subroutine write_best_final ()
- subroutine write_population (to_file)
- subroutine write_termination_case (case)
- subroutine set_optional ()

16.77.1 Function/Subroutine Documentation

16.77.1.1 subroutine sce::set_optional ( ) [private]

Definition at line 1479 of file mo_sce.f90.

Referenced by mo_sce::sce().

Here is the caller graph for this function:

16.77.1.2 subroutine sce::write_best_final ( ) [private]

Definition at line 1395 of file mo_sce.f90.

Referenced by mo_sce::sce().

Here is the caller graph for this function:

16.77.1.3 subroutine sce::write_best_intermediate (logical, intent(in) *to_file*)

Definition at line 1363 of file mo_sce.f90.

Referenced by mo_sce::sce().

Here is the caller graph for this function:

16.77.1.4 subroutine sce::write_population (logical, intent(in) *to_file*) [private]

Definition at line 1411 of file mo_sce.f90.

Referenced by mo_sce::sce().

Here is the caller graph for this function:

16.77.1.5 subroutine sce::write_termination_case (integer(i4), intent(in) *case*) [private]

Definition at line 1446 of file mo_sce.f90.

Referenced by mo_sce::sce().

Here is the caller graph for this function:

16.78 mo_set_netcdf_outputs.f90 File Reference

Data Types

- module mo_set_netcdf_outputs

  Defines the structure of the netCDF to write the output in.

16.79 mo_snow_accum_melt.f90 File Reference

Data Types

- module mo_snow_accum_melt

  Snow melting and accumulation.

16.80 mo_soil_moisture.f90 File Reference

Data Types

- module mo_soil_moisture

  Soil moisture of the different layers.

16.81 mo_spatial_agg_disagg_forcing.f90 File Reference

Data Types

- module mo_spatial_agg_disagg_forcing

  Spatial aggegation or disaggregation of meteorological input data.
- 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.

16.82 mo_spatialsimilarity.f90 File Reference

Data Types

- module mo_spatialsimilarity

  Routines for bias insensitive comparison of spatial patterns.
- interface mo_spatialsimilarity::nndv

  Calculates the number of neighboring dominating values, a measure for spatial dissimilarity.
- interface mo_spatialsimilarity::pd

  Calculates pattern dissimilarity (PD) measure.

16.83 mo_standard_score.f90 File Reference

Data Types

- module mo_standard_score

  Routines for calculating the normalization (anomaly)/standard score/z score and the deseasonalized (standard score on monthly basis) values of a time series.
- interface mo_standard_score::standard_score

  Calculates the standard score / normalization (anomaly) / z-score.
- interface mo_standard_score::classified_standard_score

  Calculates the classified standard score (e.g. classes are months).

16.84 mo_startup.f90 File Reference

Data Types

- module mo_startup

  Startup procedures for mHM.

16.85 mo_string_utils.f90 File Reference

Data Types

- module mo_string_utils

  String utilities.
- interface mo_string_utils::num2str

  Convert to string.
- interface mo_string_utils::numarray2str

  Convert to string.

16.86 mo_template.f90 File Reference

Data Types

- module mo_template

  Template for future module developments.
- interface mo_template::mean

  The average.

16.87 mo_temporal_aggregation.f90 File Reference

Data Types

- module mo_temporal_aggregation

  Temporal aggregation for time series (averaging)
- interface mo_temporal_aggregation::day2mon_average

  Day-to-month average (day2mon_average)
- interface mo_temporal_aggregation::hour2day_average

  Hour-to-day average (hour2day_average)

16.88 mo_temporal_disagg_forcing.f90 File Reference

Data Types

- module mo_temporal_disagg_forcing

  Temporal disaggregation of daily input values.

16.89 mo_timer.f90 File Reference

Data Types

- module mo_timer

  Timing routines.

16.90 mo_utils.f90 File Reference

Data Types

- module mo_utils

  General utilities for the CHS library.
- interface mo_utils::equal

  Comparison of real values.
- interface mo_utils::notequal
- interface mo_utils::greaterequal
- interface mo_utils::lesserequal
- interface mo_utils::eq
- interface mo_utils::ne
- interface mo_utils::ge
- interface mo_utils::le
- interface mo_utils::is_finite

  .true. if not IEEE Inf, IEEE NaN, nor IEEE Inf nor IEEE NaN, respectively.
- interface mo_utils::is_nan
- interface mo_utils::is_normal
- interface mo_utils::locate

  Find closest values in a monotonic series, returns the indexes.
- interface mo_utils::swap

  Swap to values or two elements in array.
- interface mo_utils::special_value

  Special IEEE values.

16.91 mo_write_ascii.f90 File Reference

Data Types

- module mo_write_ascii

  Module to write ascii file output.

16.92 mo_write_fluxes_states.f90 File Reference

Data Types

- module mo_write_fluxes_states

  Creates NetCDF output for different fluxes and state variables of mHM.
- interface mo_write_fluxes_states::outputvariable
- interface mo_write_fluxes_states::outputvariable
- interface mo_write_fluxes_states::outputdataset
- interface mo_write_fluxes_states::outputdataset

16.93 mo_xor4096.f90 File Reference

Data Types

- module mo_xor4096
- interface mo_xor4096::get_timeseed
- interface mo_xor4096::xor4096
- interface mo_xor4096::xor4096g

16.94 mrm_driver.f90 File Reference

Data Types

- module dummy_mrm

Functions/Subroutines

- program mrm_driver

  TODO: add description.

16.94.1 Function/Subroutine Documentation

16.94.1.1 program mrm_driver ( )

TODO: add description.

TODO: add description

Authors

Stephan Thober

Date

Jun 2018

Definition at line 15 of file mrm_driver.f90.

References mo_finish::finish(), mo_message::message(), mo_mrm_eval::mrm_eval(), mo_mrm_init::mrm_init(), mo_mrm_write::mrm_write(), mo_mrm_write::mrm_write_optifile(), mo_mrm_write::mrm_write_optinamelist(), mo_mrm_objective_function_runoff::single_objective_runoff(), mo_timer::timer_get(), mo_timer::timer_start(), mo_timer::timer_stop(), and mo_timer::timers_init().

Here is the call graph for this function:

16.95 RELEASES.md File Reference