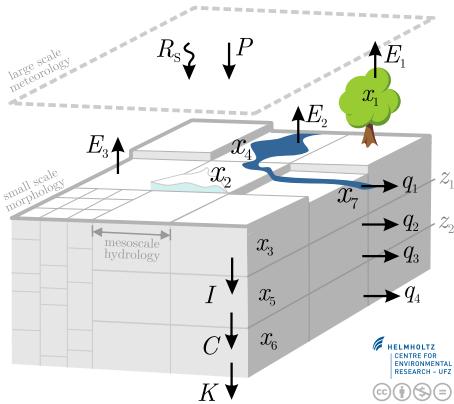


The mesoscale Hydrologic Model

mHM



Documentation for version 5.9

Edited by Luis Samaniego

in cooperation with

Johannes Brenner, Cüneyd M. Demirel, Miao Jing, Maren Kaluza,
Rohini Kumar, Ben Langenberg, Oldrich Rakovec, David Schäfer,
Martin Schrön, Robert Schweppe, Stephan Thober

Former mHM project collaborators

John Craven, Matthias Cuntz, Giovanni Dalmasso, Juliane Mai, Jude Musuuza, Vladyslav Prykhodko, Christoph Schneider, Diana Spieler, Simon Stisen, Matthias Zink

The mHM project

www.ufz.de/mhm

©2005-2018

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	Extract Data to the Time Period of Interest	21
3.2.3	Data Types	21
3.2.4	Variable Names	22
3.2.5	The Header File	22
3.2.6	NODATA values	22
3.3	Preparation of the LatLon Grid	22
3.4	Preparation of the Morphological Data	23
3.5	A possible GIS workflow	24
3.5.1	General considerations	24
3.5.2	Slope map	27
3.5.3	Aspect map	27
3.5.4	Fill DEM sinks	28
3.5.5	Flow direction and flow accumulation	28
3.5.5.1	Spatial Analyst	28
3.5.5.2	Arc Hydro Tools	29
3.5.6	Gauges map	30
3.5.7	Watershed delineation	32
3.5.8	Mask the datasets	33
3.5.9	Write the ascii grids	33
3.6	Land Cover Data	33
3.7	Table Data	34
3.7.1	The soil look-up table	34
3.7.2	The hydrogeolgy look-up table	34
3.7.3	The LAI look-up table	35
3.7.4	The gauge files	35
3.8	Post-GIS preparation	36
4	Visualizing Model Output	37
5	Calibration Options	39
5.1	Calibration of Parameters on Observations	39
5.1.1	Parameters	39
5.1.2	Methods	39
5.1.3	Functions	39
5.1.4	Data	39
5.1.5	Output	40

6 Coding and Documentation Style	41
7 The details about the test basin	45
8 Protocols for setting up a new mHM basin	47
8.1 Check for possible input data errors using mHM outputs	47
8.2 Protocol for generating a restart file	48
8.3 Protocol for determining the warm-up period for a new basin	48
9 mHM Dependencies	49
10 Publications using mHM	51
11 mHM RELEASE NOTES	55
12 Modules Index	63
12.1 Modules List	63
13 Data Type Index	67
13.1 Data Types List	67
14 File Index	71
14.1 File List	71
15 Module Documentation	75
15.1 dummy_mpr Module Reference	75
15.2 dummy_mrm Module Reference	75
15.3 mo_anneal Module Reference	75
15.3.1 Function/Subroutine Documentation	75
15.3.1.1 anneal_dp()	75
15.3.1.2 dchange_dp()	76
15.3.1.3 generate_neighborhood_weight_dp()	77
15.3.1.4 gettemperature_dp()	77
15.3.1.5 pargen_anneal_dp()	78
15.3.1.6 pargen_dds_dp()	79
15.4 mo_append Module Reference	79
15.4.1 Detailed Description	80
15.4.2 Function/Subroutine Documentation	80
15.4.2.1 append_char_3d()	80
15.4.2.2 append_char_m_m()	81
15.4.2.3 append_char_v_s()	81
15.4.2.4 append_char_v_v()	81
15.4.2.5 append_dp_3d()	81
15.4.2.6 append_dp_m_m()	81

15.4.2.7	append_dp_v_s()	81
15.4.2.8	append_dp_v_v()	81
15.4.2.9	append_i4_3d()	82
15.4.2.10	append_i4_m_m()	82
15.4.2.11	append_i4_v_s()	82
15.4.2.12	append_i4_v_v()	82
15.4.2.13	append_i8_3d()	82
15.4.2.14	append_i8_m_m()	82
15.4.2.15	append_i8_v_s()	83
15.4.2.16	append_i8_v_v()	83
15.4.2.17	append_lgt_3d()	83
15.4.2.18	append_lgt_m_m()	83
15.4.2.19	append_lgt_v_s()	83
15.4.2.20	append_lgt_v_v()	83
15.4.2.21	append_sp_3d()	83
15.4.2.22	append_sp_m_m()	84
15.4.2.23	append_sp_v_s()	84
15.4.2.24	append_sp_v_v()	84
15.4.2.25	paste_char_m_m()	84
15.4.2.26	paste_char_m_s()	84
15.4.2.27	paste_char_m_v()	84
15.4.2.28	paste_dp_m_m()	84
15.4.2.29	paste_dp_m_s()	85
15.4.2.30	paste_dp_m_v()	85
15.4.2.31	paste_i4_m_m()	85
15.4.2.32	paste_i4_m_s()	85
15.4.2.33	paste_i4_m_v()	85
15.4.2.34	paste_i8_m_m()	85
15.4.2.35	paste_i8_m_s()	86
15.4.2.36	paste_i8_m_v()	86
15.4.2.37	paste_lgt_m_m()	86
15.4.2.38	paste_lgt_m_s()	86
15.4.2.39	paste_lgt_m_v()	86
15.4.2.40	paste_sp_m_m()	86
15.4.2.41	paste_sp_m_s()	86
15.4.2.42	paste_sp_m_v()	87
15.5	mo_canopy_interc Module Reference	87
15.5.1	Detailed Description	87
15.5.2	Function/Subroutine Documentation	87
15.5.2.1	canopy_interc()	87

15.6 mo_common_constants Module Reference	89
15.6.1 Detailed Description	90
15.6.2 Variable Documentation	90
15.6.2.1 dayhours	90
15.6.2.2 daysecs	90
15.6.2.3 eps_dp	90
15.6.2.4 eps_sp	90
15.6.2.5 hoursecs	90
15.6.2.6 maxnlcovers	91
15.6.2.7 maxnobasins	91
15.6.2.8 ncolpars	91
15.6.2.9 nodata_dp	91
15.6.2.10 nodata_i4	92
15.6.2.11 p1_initstatefluxes	92
15.6.2.12 yeardays	92
15.6.2.13 yearmonths	92
15.6.2.14 yearmonths_i4	92
15.7 mo_common_file Module Reference	92
15.7.1 Detailed Description	93
15.7.2 Variable Documentation	93
15.7.2.1 file_config	93
15.7.2.2 file_dem	93
15.7.2.3 uconfig	93
15.7.2.4 udem	94
15.7.2.5 ulcoverclass	94
15.8 mo_common_functions Module Reference	94
15.8.1 Detailed Description	94
15.8.2 Function/Subroutine Documentation	94
15.8.2.1 in_bound()	94
15.9 mo_common_mhm_mrm_file Module Reference	95
15.9.1 Detailed Description	95
15.9.2 Variable Documentation	95
15.9.2.1 file_opti	96
15.9.2.2 file_opti_nml	96
15.9.2.3 uopti	96
15.9.2.4 uopti_nml	96
15.10 mo_common_mhm_mrm_read_config Module Reference	96
15.10.1 Detailed Description	96
15.10.2 Function/Subroutine Documentation	97
15.10.2.1 check_optimization_settings()	97

15.10.2.2 common_check_resolution()	98
15.10.2.3 common_mhm_mrm_read_config()	99
15.11 mo_common_mhm_mrm_variables Module Reference	100
15.11.1 Detailed Description	101
15.11.2 Variable Documentation	101
15.11.2.1 dds_r	101
15.11.2.2 dirrestartin	102
15.11.2.3 evalper	102
15.11.2.4 lcyearid	102
15.11.2.5 mcmc_error_params	102
15.11.2.6 mcmc_opti	102
15.11.2.7 mrm_coupling_mode	102
15.11.2.8 nerror_model	103
15.11.2.9 niterations	103
15.11.2.10 ntstepday	103
15.11.2.11 opti_function	103
15.11.2.12 opti_method	103
15.11.2.13 optimize	103
15.11.2.14 optimize_restart	103
15.11.2.15 read_restart	104
15.11.2.16 readper	104
15.11.2.17 resolutionrouting	104
15.11.2.18 sa_temp	104
15.11.2.19 sce_ngs	104
15.11.2.20 sce_npg	104
15.11.2.21 sce_nps	105
15.11.2.22 seed	105
15.11.2.23 simper	105
15.11.2.24 timestep	105
15.11.2.25 warmingdays	105
15.11.2.26 warmper	105
15.12 mo_common_read_config Module Reference	106
15.12.1 Detailed Description	106
15.12.2 Function/Subroutine Documentation	106
15.12.2.1 common_read_config()	106
15.12.2.2 set_land_cover_scenes_id()	107
15.13 mo_common_read_data Module Reference	108
15.13.1 Detailed Description	109
15.13.2 Function/Subroutine Documentation	109
15.13.2.1 read_dem()	109

15.13.2.2 <code>read_lcover()</code>	110
15.14 <code>mo_common_restart</code> Module Reference	111
15.14.1 Detailed Description	111
15.14.2 Function/Subroutine Documentation	111
15.14.2.1 <code>read_grid_info()</code>	111
15.14.2.2 <code>write_grid_info()</code>	113
15.15 <code>mo_common_variables</code> Module Reference	114
15.15.1 Detailed Description	115
15.15.2 Variable Documentation	115
15.15.2.1 <code>alma_convention</code>	116
15.15.2.2 <code>contact</code>	116
15.15.2.3 <code>conventions</code>	116
15.15.2.4 <code>dircommonfiles</code>	116
15.15.2.5 <code>dirconfigout</code>	116
15.15.2.6 <code>dirlcover</code>	116
15.15.2.7 <code>dirmorpho</code>	116
15.15.2.8 <code>dirout</code>	117
15.15.2.9 <code>dirrestartout</code>	117
15.15.2.10 <code>filelatlon</code>	117
15.15.2.11 <code>global_parameters</code>	117
15.15.2.12 <code>global_parameters_name</code>	117
15.15.2.13 <code>history</code>	117
15.15.2.14 <code>flag_coordinate_sys</code>	117
15.15.2.15 <code>0_basin</code>	118
15.15.2.16 <code>0_elev</code>	118
15.15.2.17 <code>0_l1_remap</code>	118
15.15.2.18 <code>0_lcover</code>	118
15.15.2.19 <code>c_year_end</code>	118
15.15.2.20 <code>c_year_start</code>	118
15.15.2.21 <code>lcfilename</code>	119
15.15.2.22 <code>level0</code>	119
15.15.2.23 <code>level1</code>	119
15.15.2.24 <code>mhm_details</code>	119
15.15.2.25 <code>nbasins</code>	120
15.15.2.26 <code>nlcoverScene</code>	120
15.15.2.27 <code>nprocesses</code>	120
15.15.2.28 <code>uniqueL0basins</code>	120
15.15.2.29 <code>processmatrix</code>	120
15.15.2.30 <code>project_details</code>	121
15.15.2.31 <code>resolutionhydrology</code>	121

15.15.2.32	setup_description	121
15.15.2.33	simulation_type	121
15.15.2.34	write_restart	121
15.16	mo_constants Module Reference	121
15.16.1	Detailed Description	124
15.16.2	Variable Documentation	124
15.16.2.1	cp0_dp	124
15.16.2.2	cp0_sp	124
15.16.2.3	dayhours	124
15.16.2.4	daysecs	124
15.16.2.5	deg2rad_dp	124
15.16.2.6	deg2rad_sp	125
15.16.2.7	euler	125
15.16.2.8	euler_d	125
15.16.2.9	gravity_dp	125
15.16.2.10	gravity_sp	125
15.16.2.11	nerr	125
15.16.2.12	min	125
15.16.2.13	nnml	125
15.16.2.14	nhout	126
15.16.2.15	p0_dp	126
15.16.2.16	p0_sp	126
15.16.2.17	pi	126
15.16.2.18	pi_d	126
15.16.2.19	pi_dp	126
15.16.2.20	pi_sp	126
15.16.2.21	pio2	127
15.16.2.22	pio2_d	127
15.16.2.23	pio2_dp	127
15.16.2.24	pio2_sp	127
15.16.2.25	psychro_dp	127
15.16.2.26	psychro_sp	127
15.16.2.27	rad2deg_dp	127
15.16.2.28	rad2deg_sp	128
15.16.2.29	radiusearth_dp	128
15.16.2.30	radiusearth_sp	128
15.16.2.31	rho0_dp	128
15.16.2.32	rho0_sp	128
15.16.2.33	secday_dp	128
15.16.2.34	secday_sp	128

15.16.2.35	sigma_dp	128
15.16.2.36	sigma_sp	129
15.16.2.37	solarconst_dp	129
15.16.2.38	solarconst_sp	129
15.16.2.39	specheatet_dp	129
15.16.2.40	specheatet_sp	129
15.16.2.41	sqrt2	129
15.16.2.42	sqrt2_d	129
15.16.2.43	sqrt2_dp	130
15.16.2.44	sqrt2_sp	130
15.16.2.45	0_dp	130
15.16.2.46	0_sp	130
15.16.2.47	twopi	130
15.16.2.48	wopi_d	130
15.16.2.49	wopi_dp	130
15.16.2.50	wopi_sp	131
15.16.2.51	twothird_dp	131
15.16.2.52	twothird_sp	131
15.16.2.53	yeardays	131
15.16.2.54	yearmonths	131
15.17	mo_corr Module Reference	131
15.17.1	Function/Subroutine Documentation	132
15.17.1.1	arth_dp()	132
15.17.1.2	arth_i4()	133
15.17.1.3	arth_sp()	133
15.17.1.4	autocoeffk_1d_dp()	133
15.17.1.5	autocoeffk_1d_sp()	133
15.17.1.6	autocoeffk_dp()	133
15.17.1.7	autocoeffk_sp()	133
15.17.1.8	autocorr_1d_dp()	134
15.17.1.9	autocorr_1d_sp()	134
15.17.1.10	autocorr_dp()	134
15.17.1.11	autocorr_sp()	134
15.17.1.12	corr_dp()	134
15.17.1.13	corr_sp()	134
15.17.1.14	crosscoeffk_dp()	135
15.17.1.15	crosscoeffk_sp()	135
15.17.1.16	crosscorr_dp()	135
15.17.1.17	crosscorr_sp()	135
15.17.1.18	four1_dp()	135

15.17.1.19four1_sp()	135
15.17.1.20fourrow_dp()	136
15.17.1.21fourrow_sp()	136
15.17.1.22realft_dp()	136
15.17.1.23realft_sp()	136
15.17.1.24swap_1d_dpc()	137
15.17.1.25swap_1d_spc()	137
15.17.1.26zroots_unity_dp()	137
15.17.1.27zroots_unity_sp()	138
15.17.2 Variable Documentation	138
15.17.2.1 npar2_arth	138
15.17.2.2 npar_arth	138
15.18mo_dds Module Reference	138
15.18.1 Detailed Description	139
15.18.2 Function/Subroutine Documentation	139
15.18.2.1 dds()	139
15.18.2.2 mdds()	141
15.18.2.3 neigh_value()	142
15.19mo_errormeasures Module Reference	142
15.19.1 Function/Subroutine Documentation	144
15.19.1.1 bias_dp_1d()	144
15.19.1.2 bias_dp_2d()	144
15.19.1.3 bias_dp_3d()	144
15.19.1.4 bias_sp_1d()	145
15.19.1.5 bias_sp_2d()	145
15.19.1.6 bias_sp_3d()	145
15.19.1.7 kge_dp_1d()	145
15.19.1.8 kge_dp_2d()	145
15.19.1.9 kge_dp_3d()	145
15.19.1.10kge_sp_1d()	145
15.19.1.11kge_sp_2d()	146
15.19.1.12kge_sp_3d()	146
15.19.1.13kgenocorr_dp_1d()	146
15.19.1.14kgenocorr_dp_2d()	146
15.19.1.15kgenocorr_dp_3d()	146
15.19.1.16kgenocorr_sp_1d()	146
15.19.1.17kgenocorr_sp_2d()	147
15.19.1.18kgenocorr_sp_3d()	147
15.19.1.19nnse_dp_1d()	147
15.19.1.20nnse_dp_2d()	147

15.19.1.21nnse_dp_3d()	147
15.19.1.22nnse_sp_1d()	147
15.19.1.23nnse_sp_2d()	148
15.19.1.24nnse_sp_3d()	148
15.19.1.25mae_dp_1d()	148
15.19.1.26mae_dp_2d()	148
15.19.1.27mae_dp_3d()	149
15.19.1.28mae_sp_1d()	149
15.19.1.29mae_sp_2d()	150
15.19.1.30mae_sp_3d()	150
15.19.1.31mse_dp_1d()	150
15.19.1.32mse_dp_2d()	151
15.19.1.33mse_dp_3d()	152
15.19.1.34mse_sp_1d()	152
15.19.1.35mse_sp_2d()	153
15.19.1.36mse_sp_3d()	154
15.19.1.37hse_dp_1d()	154
15.19.1.38hse_dp_2d()	155
15.19.1.39hse_dp_3d()	155
15.19.1.40hse_sp_1d()	155
15.19.1.41hse_sp_2d()	155
15.19.1.42hse_sp_3d()	155
15.19.1.43mse_dp_1d()	155
15.19.1.44mse_dp_2d()	156
15.19.1.45mse_dp_3d()	156
15.19.1.46mse_sp_1d()	157
15.19.1.47mse_sp_2d()	157
15.19.1.48mse_sp_3d()	157
15.19.1.49sae_dp_1d()	158
15.19.1.50sae_dp_2d()	158
15.19.1.51sae_dp_3d()	159
15.19.1.52sae_sp_1d()	160
15.19.1.53sae_sp_2d()	160
15.19.1.54sae_sp_3d()	161
15.19.1.55sse_dp_1d()	162
15.19.1.56sse_dp_2d()	162
15.19.1.57sse_dp_3d()	163
15.19.1.58sse_sp_1d()	164
15.19.1.59sse_sp_2d()	164
15.19.1.60sse_sp_3d()	165

15.19.1.61wnse_dp_1d()	166
15.19.1.62wnse_dp_2d()	166
15.19.1.63wnse_dp_3d()	166
15.19.1.64wnse_sp_1d()	166
15.19.1.65wnse_sp_2d()	166
15.19.1.66wnse_sp_3d()	167
15.20mo_file Module Reference	167
15.20.1 Detailed Description	167
15.20.2 Variable Documentation	167
15.20.2.1 file_defoutput	168
15.20.2.2 file_main	168
15.20.2.3 file_namelist_mhm	168
15.20.2.4 file_namelist_mhm_param	168
15.20.2.5 udefoutput	168
15.20.2.6 unamelist_mhm	168
15.20.2.7 unamelist_mhm_param	168
15.20.2.8 utws	169
15.20.2.9 version	169
15.20.2.10version_date	169
15.21mo_finish Module Reference	169
15.21.1 Detailed Description	169
15.21.2 Function/Subroutine Documentation	169
15.21.2.1 finish()	170
15.22mo_global_variables Module Reference	171
15.22.1 Detailed Description	172
15.22.2 Variable Documentation	172
15.22.2.1 basin_avg_tws_obs	173
15.22.2.2 basin_avg_tws_sim	173
15.22.2.3 dirabsvappressure	173
15.22.2.4 direvapotranspiration	173
15.22.2.5 dirmaxtemperature	173
15.22.2.6 dirmintemperature	173
15.22.2.7 dirnetradiation	173
15.22.2.8 dirneutrons	174
15.22.2.9 dirprecipitation	174
15.22.2.10dirreferenceet	174
15.22.2.11dirsoil_moisture	174
15.22.2.12dirtemperature	174
15.22.2.13dirwindspeed	174
15.22.2.14evap_coeff	174

15.22.2.15	day_pet	175
15.22.2.16	day_prec	175
15.22.2.17	day_temp	175
15.22.2.18	filetw	175
15.22.2.19	night_pet	175
15.22.2.20	night_prec	175
15.22.2.21	night_temp	175
15.22.2.22	npusformat_meteo_forcings	175
15.22.2.23	1_absvapress	176
15.22.2.24	1_aetcanopy	176
15.22.2.25	1_aetsealed	176
15.22.2.26	1_aetsoil	176
15.22.2.27	1_baseflow	176
15.22.2.28	1_et	176
15.22.2.29	1_et_mask	177
15.22.2.30	1_fastrunoff	177
15.22.2.31	1_infilsoil	177
15.22.2.32	1_inter	177
15.22.2.33	1_melt	177
15.22.2.34	1_netrad	177
15.22.2.35	1_neutrons	177
15.22.2.36	1_neutronsdata	178
15.22.2.37	1_neutronsdata_mask	178
15.22.2.38	1_percol	178
15.22.2.39	1_pet	178
15.22.2.40	1_pet_calc	178
15.22.2.41	1_pet_weights	178
15.22.2.42	1_pre	178
15.22.2.43	1_pre_weights	179
15.22.2.44	1_preeffect	179
15.22.2.45	1_rain	179
15.22.2.46	1_runoffseal	179
15.22.2.47	1_satstw	179
15.22.2.48	1_sealstw	179
15.22.2.49	1_slowrunoff	179
15.22.2.50	1_sm	180
15.22.2.51	1_sm_mask	180
15.22.2.52	1_snow	180
15.22.2.53	1_snowpack	180
15.22.2.54	1_soilmoist	180

15.22.2.551_temp	180
15.22.2.561_temp_weights	180
15.22.2.571_throughfall	181
15.22.2.581_tmax	181
15.22.2.591_tmin	181
15.22.2.601_total_runoff	181
15.22.2.611_unsatstw	181
15.22.2.621_windspeed	181
15.22.2.63level2	181
15.22.2.64neutron_integral_afast	182
15.22.2.65measperday_tws	182
15.22.2.66soilhorizons_sm_input	182
15.22.2.67ntimesteps_l1_et	182
15.22.2.68ntimesteps_l1_neurons	182
15.22.2.69ntimesteps_l1_sm	182
15.22.2.70outputflxstate	182
15.22.2.71read_meteo_weights	183
15.22.2.72routingstates	183
15.22.2.73timestep_et_input	183
15.22.2.74timestep_model_inputs	183
15.22.2.75timestep_model_outputs	183
15.22.2.76timestep_neurons_input	183
15.22.2.77timestep_sm_input	183
15.23mo_grid Module Reference	184
15.23.1 Detailed Description	184
15.23.2 Function/Subroutine Documentation	184
15.23.2.1 calculate_grid_properties()	184
15.23.2.2 geocoordinates()	186
15.23.2.3 init_lowres_level()	186
15.23.2.4 io_grid_setup()	188
15.23.2.5 mapcoordinates()	188
15.23.2.6 set_basin_indices()	189
15.24mo_init_states Module Reference	190
15.24.1 Detailed Description	190
15.24.2 Function/Subroutine Documentation	190
15.24.2.1 variables_alloc()	191
15.24.2.2 variables_default_init()	191
15.25mo_julian Module Reference	192
15.25.1 Detailed Description	194
15.25.2 Function/Subroutine Documentation	194

15.25.2.1 caldat()	194
15.25.2.2 caldat360()	196
15.25.2.3 caldat365()	197
15.25.2.4 caldatjulian()	197
15.25.2.5 date2dec()	199
15.25.2.6 date2dec360()	200
15.25.2.7 date2dec365()	201
15.25.2.8 date2decjulian()	203
15.25.2.9 dec2date()	204
15.25.2.10 dec2date360()	206
15.25.2.11 dec2date365()	208
15.25.2.12 dec2datejulian()	209
15.25.2.13 julday()	211
15.25.2.14 julday360()	212
15.25.2.15 julday365()	213
15.25.2.16 juldayjulian()	214
15.25.2.17 ndays()	216
15.25.2.18 ndyin()	217
15.25.2.19 selectcalendar()	218
15.25.2.20 setcalendarinteger()	219
15.25.2.21 setcalendarstring()	220
15.25.3 Variable Documentation	220
15.25.3.1 calendar	220
15.26 mo_kind Module Reference	221
15.26.1 Detailed Description	221
15.26.2 Variable Documentation	222
15.26.2.1 dp	222
15.26.2.2 dpc	222
15.26.2.3 i1	222
15.26.2.4 i2	222
15.26.2.5 i4	222
15.26.2.6 i8	223
15.26.2.7 lgt	223
15.26.2.8 sp	223
15.26.2.9 spc	223
15.27 mo_linfit Module Reference	224
15.27.1 Detailed Description	224
15.27.2 Function/Subroutine Documentation	224
15.27.2.1 linfit_dp()	224
15.27.2.2 linfit_sp()	224

15.28	mo_mcmc Module Reference	225
15.28.1	Detailed Description	225
15.28.2	Function/Subroutine Documentation	225
15.28.2.1	generatenewparameterset_dp()	226
15.28.2.2	mcmc_dp()	226
15.28.2.3	mcmc_stddev_dp()	227
15.28.2.4	pargen_dp()	228
15.28.2.5	pargennorm_dp()	228
15.29	mo_message Module Reference	228
15.29.1	Detailed Description	228
15.29.2	Function/Subroutine Documentation	229
15.29.2.1	message()	229
15.29.3	Variable Documentation	230
15.29.3.1	message_text	230
15.30	mo_meteo_forcings Module Reference	230
15.30.1	Detailed Description	231
15.30.2	Function/Subroutine Documentation	231
15.30.2.1	chunk_config()	231
15.30.2.2	chunk_size()	232
15.30.2.3	is_read()	234
15.30.2.4	meteo_forcings_wrapper()	235
15.30.2.5	meteo_weights_wrapper()	236
15.30.2.6	prepare_meteo_forcings_data()	237
15.31	mo_mhm Module Reference	238
15.31.1	Detailed Description	238
15.31.2	Function/Subroutine Documentation	239
15.31.2.1	mhm()	239
15.32	mo_mhm_constants Module Reference	244
15.32.1	Detailed Description	245
15.32.2	Variable Documentation	245
15.32.2.1	c1_initstatesm	245
15.32.2.2	cosmic_alpha	246
15.32.2.3	cosmic_bd	246
15.32.2.4	cosmic_l1	246
15.32.2.5	cosmic_l2	246
15.32.2.6	cosmic_l3	246
15.32.2.7	cosmic_l4	246
15.32.2.8	cosmic_n	246
15.32.2.9	cosmic_vwclat	246
15.32.2.10	desilets_a0	247

15.32.2.11desilets_a1	247
15.32.2.12desilets_a2	247
15.32.2.13duffiedelta1	247
15.32.2.14duffiedelta2	247
15.32.2.15duffiedr	247
15.32.2.16h2odens	247
15.32.2.17harsamconst	248
15.32.2.18noutflxstate	248
15.32.2.19p2_initstatefluxes	248
15.32.2.20p3_initstatefluxes	248
15.32.2.21p4_initstatefluxes	248
15.32.2.22p5_initstatefluxes	248
15.32.2.23satpressureslope1	248
15.32.2.24stboltzmann	248
15.32.2.25etens_c1	249
15.32.2.26etens_c2	249
15.32.2.27etens_c3	249
15.33mo_mhm_eval Module Reference	249
15.33.1 Detailed Description	249
15.33.2 Function/Subroutine Documentation	249
15.33.2.1 mhm_eval()	249
15.34mo_mhm_read_config Module Reference	253
15.34.1 Detailed Description	253
15.34.2 Function/Subroutine Documentation	253
15.34.2.1 mhm_read_config()	253
15.35mo_moment Module Reference	256
15.35.1 Function/Subroutine Documentation	257
15.35.1.1 absdev_dp()	257
15.35.1.2 absdev_sp()	257
15.35.1.3 average_dp()	257
15.35.1.4 average_sp()	257
15.35.1.5 central_moment_dp()	257
15.35.1.6 central_moment_sp()	258
15.35.1.7 central_moment_var_dp()	258
15.35.1.8 central_moment_var_sp()	258
15.35.1.9 correlation_dp()	258
15.35.1.10correlation_sp()	258
15.35.1.11covariance_dp()	258
15.35.1.12covariance_sp()	258
15.35.1.13kurtosis_dp()	259

15.35.1.14	kurtosis_sp()	259
15.35.1.15	mean_dp()	259
15.35.1.16	mean_sp()	259
15.35.1.17	mixed_central_moment_dp()	259
15.35.1.18	mixed_central_moment_sp()	259
15.35.1.19	mixed_central_moment_var_dp()	260
15.35.1.20	mixed_central_moment_var_sp()	260
15.35.1.21	moment_dp()	260
15.35.1.22	moment_sp()	260
15.35.1.23	skewness_dp()	260
15.35.1.24	skewness_sp()	261
15.35.1.25	stddev_dp()	261
15.35.1.26	stddev_sp()	261
15.35.1.27	variance_dp()	261
15.35.1.28	variance_sp()	261
15.36	mo_mpr_constants Module Reference	261
15.36.1	Detailed Description	262
15.36.2	Variable Documentation	263
15.36.2.1	bulkdens_orgmatter	263
15.36.2.2	c1_initstatesm	263
15.36.2.3	field_cap_c1	263
15.36.2.4	field_cap_c2	263
15.36.2.5	karman	263
15.36.2.6	ks_c	263
15.36.2.7	lai_factor_surfresi	263
15.36.2.8	lai_offset_surfresi	264
15.36.2.9	max_surfresist	264
15.36.2.10	maxgeounit	264
15.36.2.11	maxnosoilhorizons	264
15.36.2.12	nlcover_class	264
15.36.2.13	p2_initstatefluxes	264
15.36.2.14	p3_initstatefluxes	264
15.36.2.15	p4_initstatefluxes	265
15.36.2.16	p5_initstatefluxes	265
15.36.2.17	pwp_c	265
15.36.2.18	pwp_matpot_theta	265
15.36.2.19	genuchten_sandtresh	265
15.36.2.20	genuchten_c1	265
15.36.2.21	genuchten_c10	265
15.36.2.22	genuchten_c11	265

15.36.2.23vgenuchtenn_c12	266
15.36.2.24vgenuchtenn_c13	266
15.36.2.25vgenuchtenn_c14	266
15.36.2.26vgenuchtenn_c15	266
15.36.2.27vgenuchtenn_c16	266
15.36.2.28vgenuchtenn_c17	266
15.36.2.29vgenuchtenn_c18	266
15.36.2.30vgenuchtenn_c2	267
15.36.2.31vgenuchtenn_c3	267
15.36.2.32vgenuchtenn_c4	267
15.36.2.33vgenuchtenn_c5	267
15.36.2.34vgenuchtenn_c6	267
15.36.2.35vgenuchtenn_c7	267
15.36.2.36vgenuchtenn_c8	267
15.36.2.37vgenuchtenn_c9	267
15.36.2.38windmeasheight	268
15.37mo_mpr_eval Module Reference	268
15.37.1 Detailed Description	268
15.37.2 Function/Subroutine Documentation	268
15.37.2.1 mpr_eval()	268
15.38mo_mpr_file Module Reference	271
15.38.1 Detailed Description	272
15.38.2 Variable Documentation	272
15.38.2.1 file_aspect	272
15.38.2.2 file_geolut	272
15.38.2.3 file_hydrogeoclass	272
15.38.2.4 file_laiclass	273
15.38.2.5 file_lailut	273
15.38.2.6 file_main	273
15.38.2.7 file_meteo_binary_end	273
15.38.2.8 file_meteo_header	273
15.38.2.9 file_namelist_mpr	273
15.38.2.10file_namelist_mpr_param	273
15.38.2.11file_slope	274
15.38.2.12file_soil_database	274
15.38.2.13file_soil_database_1	274
15.38.2.14file_soilclass	274
15.38.2.15uaspect	274
15.38.2.16ugeolut	274
15.38.2.17uhydrogeoclass	275

15.38.2.18laiaclass	275
15.38.2.19lailut	275
15.38.2.20umeteo	275
15.38.2.21umeteo_header	275
15.38.2.22unamelist_mpr	275
15.38.2.23unamelist_mpr_param	275
15.38.2.24uslope	276
15.38.2.25usoil_database	276
15.38.2.26usoilclass	276
15.38.2.27version	276
15.38.2.28version_date	276
15.39mo_mpr_global_variables Module Reference	276
15.39.1 Detailed Description	278
15.39.2 Variable Documentation	278
15.39.2.1 c2tstu	278
15.39.2.2 dirgridded_lai	278
15.39.2.3 fracsealed_cityarea	278
15.39.2.4 geounitkar	278
15.39.2.5 geounitlist	278
15.39.2.6 horizontdepth_mhm	279
15.39.2.7 iflag_soildb	279
15.39.2.8 inputformat_gridded_lai	279
15.39.2.9 l0_asp	279
15.39.2.100_geounit	279
15.39.2.110_gridded_lai	279
15.39.2.120_slope	279
15.39.2.130_slope_emp	280
15.39.2.140_soilid	280
15.39.2.151_aeroresist	280
15.39.2.161_alpha	280
15.39.2.171_degday	280
15.39.2.181_degdayinc	280
15.39.2.191_degdaymax	281
15.39.2.201_degdaynopre	281
15.39.2.211_fasp	281
15.39.2.221_froots	281
15.39.2.231_fsealed	281
15.39.2.241_harsamcoeff	281
15.39.2.251_jarvis_thresh_c1	281
15.39.2.261_karstloss	282

15.39.2.271_kbaseflow	282
15.39.2.281_kfastflow	282
15.39.2.291_kperco	282
15.39.2.301_kslowflow	282
15.39.2.311_maxinter	282
15.39.2.321_petlaicorfactor	282
15.39.2.331_prietaryalpha	283
15.39.2.341_sealedthresh	283
15.39.2.351_soilmoistexp	283
15.39.2.361_soilmoistfc	283
15.39.2.371_soilmoistsat	283
15.39.2.381_surfresist	283
15.39.2.391_tempthresh	284
15.39.2.401_unsatthresh	284
15.39.2.411_wiltingpoint	284
15.39.2.42ailut	284
15.39.2.43aiper	284
15.39.2.44aiunitlist	284
15.39.2.45ngeounits	284
15.39.2.46lai	284
15.39.2.47nlaiclass	285
15.39.2.48nsoilhorizons_mhm	285
15.39.2.49nsoiltypes	285
15.39.2.50soildb	285
15.39.2.51tillagedepth	285
15.39.2.52timestep_lai_input	285
15.40mo_mpr_pet Module Reference	286
15.40.1 Detailed Description	286
15.40.2 Function/Subroutine Documentation	286
15.40.2.1 bulksurface_resistance()	286
15.40.2.2 pet_correctbyasp()	287
15.40.2.3 pet_correctbylai()	288
15.40.2.4 priestley_taylor_alpha()	290
15.41mo_mpr_read_config Module Reference	291
15.41.1 Detailed Description	292
15.41.2 Function/Subroutine Documentation	292
15.41.2.1 mpr_read_config()	292
15.42mo_mpr_restart Module Reference	293
15.42.1 Detailed Description	294
15.42.2 Function/Subroutine Documentation	294

15.42.2.1 unpack_field_and_write_1d_dp()	294
15.42.2.2 unpack_field_and_write_1d_i4()	294
15.42.2.3 unpack_field_and_write_2d_dp()	295
15.42.2.4 unpack_field_and_write_3d_dp()	295
15.42.2.5 write_eff_params()	295
15.42.2.6 write_mpr_restart_files()	296
15.43 mo_mpr_runoff Module Reference	297
15.43.1 Detailed Description	298
15.43.2 Function/Subroutine Documentation	298
15.43.2.1 mpr_runoff()	298
15.44 mo_mpr_smhorizons Module Reference	300
15.44.1 Detailed Description	300
15.44.2 Function/Subroutine Documentation	300
15.44.2.1 mpr_smhorizons()	300
15.45 mo_mpr_soilmoist Module Reference	302
15.45.1 Detailed Description	303
15.45.2 Function/Subroutine Documentation	303
15.45.2.1 field_cap()	303
15.45.2.2 genuchten()	304
15.45.2.3 hydro_cond()	305
15.45.2.4 mpr_sm()	306
15.45.2.5 pwp()	308
15.46 mo_mpr_startup Module Reference	309
15.46.1 Detailed Description	309
15.46.2 Function/Subroutine Documentation	310
15.46.2.1 init_eff_params()	310
15.46.2.2 l0_check_input()	311
15.46.2.3 l0_variable_init()	312
15.46.2.4 mpr_initialize()	313
15.47 mo_mrm_constants Module Reference	314
15.47.1 Detailed Description	315
15.47.2 Variable Documentation	315
15.47.2.1 deltah	315
15.47.2.2 given_ts	315
15.47.2.3 maxnogauges	315
15.47.2.4 noutflxstate	315
15.47.2.5 nroutingstates	316
15.47.2.6 rout_space_weight	316
15.48 mo_mrm_eval Module Reference	316
15.48.1 Detailed Description	316

15.48.2 Function/Subroutine Documentation	316
15.48.2.1 mrm_eval()	316
15.49 mo_mrm_file Module Reference	318
15.49.1 Detailed Description	319
15.49.2 Variable Documentation	320
15.49.2.1 file_config	320
15.49.2.2 file_daily_discharge	320
15.49.2.3 file_defoutput	320
15.49.2.4 file_facc	320
15.49.2.5 file_fdir	320
15.49.2.6 file_gaugeloc	320
15.49.2.7 file_main	321
15.49.2.8 file_mrm_output	321
15.49.2.9 file_namelist_mrm	321
15.49.2.10 file_namelist_param_mrm	321
15.49.2.11 file_ncfile_discharge	321
15.49.2.12 uconfig	321
15.49.2.13 udaily_discharge	321
15.49.2.14 udefoutput	322
15.49.2.15 udischarge	322
15.49.2.16 ufacc	322
15.49.2.17 ufdir	322
15.49.2.18 ugaugeloc	322
15.49.2.19 unamelist_mrm	322
15.49.2.20 unamelist_param_mrm	323
15.49.2.21 version	323
15.49.2.22 version_date	323
15.50 mo_mrm_global_variables Module Reference	323
15.50.1 Detailed Description	324
15.50.2 Variable Documentation	325
15.50.2.1 basin_mrm	325
15.50.2.2 dirgauges	325
15.50.2.3 dirtotalrunoff	325
15.50.2.4 filenametotalrunoff	325
15.50.2.5 gauge	325
15.50.2.6 inflowgauge	325
15.50.2.7 is_start	326
15.50.2.8 l0_dracell	326
15.50.2.9 l0_drasc	326
15.50.2.100 facc	326

15.50.2.110_fdir	326
15.50.2.120_floodplain	326
15.50.2.130_gaugeloc	327
15.50.2.140_inflowgaugeloc	327
15.50.2.150_l11_remap	327
15.50.2.160_streamnet	327
15.50.2.1711_afloodplain	327
15.50.2.1811_c1	327
15.50.2.1911_c2	327
15.50.2.2011_colout	328
15.50.2.2111_fcol	328
15.50.2.2211_fdir	328
15.50.2.2311_fromn	328
15.50.2.2411_frow	328
15.50.2.2511_k	328
15.50.2.2611_l1_id	329
15.50.2.2711_label	329
15.50.2.2811_length	329
15.50.2.2911_netperm	329
15.50.2.3011_nlinkfracpimp	329
15.50.2.3111_noutlets	329
15.50.2.3211_qmod	330
15.50.2.3311_qout	330
15.50.2.3411_qtin	330
15.50.2.3511_qtr	330
15.50.2.3611_rorder	330
15.50.2.3711_rowout	330
15.50.2.3811_sink	331
15.50.2.3911_slope	331
15.50.2.4011_tcol	331
15.50.2.4111_ton	331
15.50.2.4211_trow	331
15.50.2.4311_tsroute	331
15.50.2.4411_xi	331
15.50.2.4511_l11_id	332
15.50.2.4611_l11_remap	332
15.50.2.4711_total_runoff_in	332
15.50.2.48evel11	332
15.50.2.49mrm_runoff	332
15.50.2.50ngaugestotal	332

15.50.2.51ninfowgaugestotal	333
15.50.2.52nmeasperday	333
15.50.2.53outputflxstate_mrm	333
15.50.2.54timestep_model_outputs_mrm	333
15.50.2.55varnametotalrunoff	333
15.51mo_mrm_init Module Reference	333
15.51.1 Detailed Description	334
15.51.2 Function/Subroutine Documentation	334
15.51.2.1 config_output()	334
15.51.2.2 l0_check_input_routing()	335
15.51.2.3 mrm_init()	336
15.51.2.4 mrm_init_param()	339
15.51.2.5 mrm_update_param()	340
15.51.2.6 print_startup_message()	341
15.51.2.7 variables_alloc_routing()	342
15.51.2.8 variables_default_init_routing()	342
15.52mo_mrm_mpr Module Reference	343
15.52.1 Detailed Description	343
15.52.2 Function/Subroutine Documentation	344
15.52.2.1 reg_rout()	344
15.53mo_mrm_net_startup Module Reference	345
15.53.1 Detailed Description	346
15.53.2 Function/Subroutine Documentation	346
15.53.2.1 celllength()	346
15.53.2.2 get_distance_two_lat_lon_points()	347
15.53.2.3 l11_flow_direction()	348
15.53.2.4 l11_fraction_sealed_floodplain()	350
15.53.2.5 l11_l1_mapping()	351
15.53.2.6 l11_link_location()	352
15.53.2.7 l11_routing_order()	353
15.53.2.8 l11_set_drain_outlet_gauges()	354
15.53.2.9 l11_set_network_topology()	355
15.53.2.10 l11_stream_features()	356
15.53.2.11 movedownonecell()	357
15.53.2.12 moveup()	358
15.54mo_mrm_objective_function_runoff Module Reference	359
15.54.1 Detailed Description	360
15.54.2 Function/Subroutine Documentation	360
15.54.2.1 extract_runoff()	360
15.54.2.2 loglikelihood_evin2013_2()	362

15.54.2.3 loglikelihood_stddev()	364
15.54.2.4 loglikelihood_trend_no_autocorr()	365
15.54.2.5 multi_objective_ae_fdc_lsv_nse_djf()	366
15.54.2.6 multi_objective_lnnse_highflow_lnnse_lowflow()	367
15.54.2.7 multi_objective_lnnse_highflow_lnnse_lowflow_2()	369
15.54.2.8 multi_objective_nse_lnnse()	370
15.54.2.9 multi_objective_runoff()	371
15.54.2.10 objective_equal_nse_lnnse()	372
15.54.2.11 objective_kge()	374
15.54.2.12 objective_lnnse()	376
15.54.2.13 objective_multiple_gauges_kge_power6()	377
15.54.2.14 objective_nse()	378
15.54.2.15 objective_power6_nse_lnnse()	379
15.54.2.16 objective_sse()	381
15.54.2.17 objective_weighted_nse()	382
15.54.2.18 parameter_regularization()	383
15.54.2.19 single_objective_runoff()	384
15.55 mo_mrm_read_config Module Reference	385
15.55.1 Detailed Description	386
15.55.2 Function/Subroutine Documentation	386
15.55.2.1 mrm_read_config()	386
15.55.2.2 read_mrm_routing_params()	387
15.56 mo_mrm_read_data Module Reference	389
15.56.1 Detailed Description	389
15.56.2 Function/Subroutine Documentation	389
15.56.2.1 mrm_read_discharge()	389
15.56.2.2 mrm_read_l0_data()	390
15.56.2.3 mrm_read_total_runoff()	391
15.56.2.4 rotate_fdir_variable()	392
15.57 mo_mrm_restart Module Reference	393
15.57.1 Detailed Description	393
15.57.2 Function/Subroutine Documentation	394
15.57.2.1 mrm_read_restart_config()	394
15.57.2.2 mrm_read_restart_states()	395
15.57.2.3 mrm_write_restart()	396
15.58 mo_mrm_routing Module Reference	397
15.58.1 Detailed Description	398
15.58.2 Function/Subroutine Documentation	398
15.58.2.1 add_inflow()	398
15.58.2.2 l11_routing()	399

15.58.2.3 l11_runoff_acc()	401
15.58.2.4 mrm_routing()	402
15.59 mo_mrm_signatures Module Reference	405
15.59.1 Detailed Description	405
15.59.2 Function/Subroutine Documentation	406
15.59.2.1 autocorrelation()	406
15.59.2.2 flowdurationcurve()	406
15.59.2.3 limb_densities()	408
15.59.2.4 maximummonthlyflow()	409
15.59.2.5 moments()	410
15.59.2.6 peakdistribution()	412
15.59.2.7 runoffratio()	412
15.59.2.8 zeroflowratio()	414
15.60 mo_mrm_write Module Reference	414
15.60.1 Detailed Description	415
15.60.2 Function/Subroutine Documentation	415
15.60.2.1 mrm_write()	415
15.60.2.2 mrm_write_optifile()	416
15.60.2.3 mrm_write_optinamelist()	417
15.60.2.4 mrm_write_output_fluxes()	419
15.60.2.5 write_configfile()	421
15.60.2.6 write_daily_obs_sim_discharge()	422
15.60.3 Variable Documentation	423
15.60.3.1 average_counter	424
15.60.3.2 day_counter	424
15.60.3.3 month_counter	424
15.60.3.4 nc	424
15.60.3.5 year_counter	424
15.61 mo_mrm_write_fluxes_states Module Reference	424
15.61.1 Detailed Description	425
15.61.2 Function/Subroutine Documentation	425
15.61.2.1 close()	425
15.61.2.2 createoutputfile()	426
15.61.2.3 newoutputdataset()	427
15.61.2.4 newoutputvariable()	428
15.61.2.5 updatedataset()	429
15.61.2.6 updatevariable()	430
15.61.2.7 writetimestep()	431
15.61.2.8 writevariableattributes()	431
15.61.2.9 writevariabletimestep()	432

15.62mo_multi_param_reg Module Reference	433
15.62.1 Detailed Description	433
15.62.2 Function/Subroutine Documentation	433
15.62.2.1 aerodynamical_resistance()	434
15.62.2.2 baseflow_param()	435
15.62.2.3 canopy_intercept_param()	436
15.62.2.4 iper_thres_runoff()	437
15.62.2.5 karstic_layer()	438
15.62.2.6 mpr()	440
15.62.2.7 snow_acc_melt_param()	444
15.63mo_ncread Module Reference	445
15.63.1 Function/Subroutine Documentation	446
15.63.1.1 check()	446
15.63.1.2 get_info()	447
15.63.1.3 get_ncdim()	449
15.63.1.4 get_ncdimatt()	451
15.63.1.5 get_ncvar_0d_dp()	452
15.63.1.6 get_ncvar_0d_i1()	452
15.63.1.7 get_ncvar_0d_i4()	453
15.63.1.8 get_ncvar_0d_sp()	453
15.63.1.9 get_ncvar_1d_dp()	454
15.63.1.10 get_ncvar_1d_i1()	454
15.63.1.11 get_ncvar_1d_i4()	455
15.63.1.12 get_ncvar_1d_sp()	455
15.63.1.13 get_ncvar_2d_dp()	456
15.63.1.14 get_ncvar_2d_i1()	456
15.63.1.15 get_ncvar_2d_i4()	457
15.63.1.16 get_ncvar_2d_sp()	457
15.63.1.17 get_ncvar_3d_dp()	458
15.63.1.18 get_ncvar_3d_i1()	458
15.63.1.19 get_ncvar_3d_i4()	459
15.63.1.20 get_ncvar_3d_sp()	459
15.63.1.21 get_ncvar_4d_dp()	460
15.63.1.22 get_ncvar_4d_i1()	460
15.63.1.23 get_ncvar_4d_i4()	461
15.63.1.24 get_ncvar_4d_sp()	461
15.63.1.25 get_ncvar_5d_dp()	462
15.63.1.26 get_ncvar_5d_i1()	462
15.63.1.27 get_ncvar_5d_i4()	463
15.63.1.28 get_ncvar_5d_sp()	463

15.63.1.29get_ncvaratt()	464
15.63.1.30hcclose()	464
15.63.1.31ncopen()	465
15.64mo_ncwrite Module Reference	465
15.64.1 Function/Subroutine Documentation	467
15.64.1.1 check()	467
15.64.1.2 close_netcdf()	468
15.64.1.3 create_netcdf()	470
15.64.1.4 dump_netcdf_1d_dp()	471
15.64.1.5 dump_netcdf_1d_i4()	471
15.64.1.6 dump_netcdf_1d_sp()	472
15.64.1.7 dump_netcdf_2d_dp()	472
15.64.1.8 dump_netcdf_2d_i4()	473
15.64.1.9 dump_netcdf_2d_sp()	473
15.64.1.10dump_netcdf_3d_dp()	474
15.64.1.11dump_netcdf_3d_i4()	474
15.64.1.12dump_netcdf_3d_sp()	475
15.64.1.13dump_netcdf_4d_dp()	475
15.64.1.14dump_netcdf_4d_i4()	476
15.64.1.15dump_netcdf_4d_sp()	476
15.64.1.16dump_netcdf_5d_dp()	477
15.64.1.17dump_netcdf_5d_i4()	477
15.64.1.18dump_netcdf_5d_sp()	478
15.64.1.19open_netcdf()	478
15.64.1.20var2nc_1d_dp()	480
15.64.1.21var2nc_1d_i4()	481
15.64.1.22var2nc_1d_sp()	482
15.64.1.23var2nc_2d_dp()	482
15.64.1.24var2nc_2d_i4()	483
15.64.1.25var2nc_2d_sp()	484
15.64.1.26var2nc_3d_dp()	484
15.64.1.27var2nc_3d_i4()	485
15.64.1.28var2nc_3d_sp()	486
15.64.1.29var2nc_4d_dp()	486
15.64.1.30var2nc_4d_i4()	487
15.64.1.31var2nc_4d_sp()	488
15.64.1.32var2nc_5d_dp()	488
15.64.1.33var2nc_5d_i4()	489
15.64.1.34var2nc_5d_sp()	490
15.64.1.35write_dynamic_netcdf()	490

15.64.1.36write_static_netcdf()	491
15.64.2 Variable Documentation	491
15.64.2.1 dnc	491
15.64.2.2 gatt	491
15.64.2.3 maxlen	492
15.64.2.4 nattdim	492
15.64.2.5 ndims	492
15.64.2.6 ngatt	492
15.64.2.7 nmaxatt	492
15.64.2.8 nmaxdim	492
15.64.2.9 nvars	492
15.64.2.10v	492
15.65mo_netcdf Module Reference	493
15.65.1 Detailed Description	496
15.65.2 Function/Subroutine Documentation	496
15.65.2.1 check()	496
15.65.2.2 close()	497
15.65.2.3 equalncdimensions()	497
15.65.2.4 getdata1df32()	497
15.65.2.5 getdata1df64()	498
15.65.2.6 getdata1di16()	498
15.65.2.7 getdata1di32()	499
15.65.2.8 getdata1di64()	499
15.65.2.9 getdata1di8()	500
15.65.2.10getdata2df32()	500
15.65.2.11getdata2df64()	501
15.65.2.12getdata2di16()	501
15.65.2.13getdata2di32()	502
15.65.2.14getdata2di64()	502
15.65.2.15getdata2di8()	503
15.65.2.16getdata3df32()	503
15.65.2.17getdata3df64()	504
15.65.2.18getdata3di16()	504
15.65.2.19getdata3di32()	505
15.65.2.20getdata3di64()	505
15.65.2.21getdata3di8()	506
15.65.2.22getdata4df32()	506
15.65.2.23getdata4df64()	507
15.65.2.24getdata4di16()	507
15.65.2.25getdata4di32()	508

15.65.2.26getdata4di64()	508
15.65.2.27getdata4di8()	509
15.65.2.28getdata5df32()	509
15.65.2.29getdata5df64()	510
15.65.2.30getdata5di16()	510
15.65.2.31getdata5di32()	511
15.65.2.32getdata5di64()	511
15.65.2.33getdata5di8()	512
15.65.2.34getdatascalarf32()	512
15.65.2.35getdatascalarf64()	512
15.65.2.36getdatascalari16()	513
15.65.2.37getdatascalari32()	513
15.65.2.38getdatascalari64()	514
15.65.2.39getdatascalari8()	514
15.65.2.40getdimensionbyid()	515
15.65.2.41getdimensionbyname()	515
15.65.2.42getdimensionlength()	516
15.65.2.43getdimensionname()	516
15.65.2.44gettypefrominteger()	516
15.65.2.45gettypefromstring()	517
15.65.2.46getglobalattributechar()	517
15.65.2.47getglobalattributef32()	518
15.65.2.48getglobalattributef64()	518
15.65.2.49getglobalattributei16()	518
15.65.2.50getglobalattributei32()	519
15.65.2.51getglobalattributei64()	519
15.65.2.52getglobalattributei8()	520
15.65.2.53getnодimensions()	520
15.65.2.54getnvariables()	520
15.65.2.55getreaddatashape()	521
15.65.2.56getunlimiteddimension()	522
15.65.2.57getvariableattributechar()	523
15.65.2.58getvariableattributef32()	523
15.65.2.59getvariableattributef64()	524
15.65.2.60getvariableattributei16()	524
15.65.2.61getvariableattributei32()	524
15.65.2.62getvariableattributei64()	525
15.65.2.63getvariableattributei8()	525
15.65.2.64getvariablebyname()	526
15.65.2.65getvariabledimensions()	526

15.65.2.66getvariabledtype()	526
15.65.2.67getvariablefillvaluef32()	527
15.65.2.68getvariablefillvaluef64()	527
15.65.2.69getvariablefillvaluei16()	527
15.65.2.70getvariablefillvaluei32()	527
15.65.2.71getvariablefillvaluei64()	527
15.65.2.72getvariablefillvaluei8()	528
15.65.2.73getvariableids()	528
15.65.2.74getvariablename()	528
15.65.2.75getvariables()	528
15.65.2.76getvariablesshape()	529
15.65.2.77hasattribute()	529
15.65.2.78hasdimension()	529
15.65.2.79hasvariable()	529
15.65.2.80nitncdataset()	529
15.65.2.81initncdimension()	529
15.65.2.82nitncvariable()	530
15.65.2.83sdatasetunlimited()	530
15.65.2.84sunlimiteddimension()	530
15.65.2.85sunlimitedvariable()	530
15.65.2.86newncdataset()	530
15.65.2.87newncdimension()	531
15.65.2.88newncvariable()	531
15.65.2.89setdata1df32()	531
15.65.2.90setdata1df64()	531
15.65.2.91setdata1di16()	532
15.65.2.92setdata1di32()	532
15.65.2.93setdata1di64()	533
15.65.2.94setdata1di8()	533
15.65.2.95setdata2df32()	534
15.65.2.96setdata2df64()	534
15.65.2.97setdata2di16()	535
15.65.2.98setdata2di32()	535
15.65.2.99setdata2di64()	536
15.65.2.100setdata2di8()	536
15.65.2.101setdata3df32()	537
15.65.2.102setdata3df64()	537
15.65.2.103setdata3di16()	538
15.65.2.104setdata3di32()	538
15.65.2.105setdata3di64()	539

15.65.2.10 ¹⁶ Setdata3di8()	539
15.65.2.10 ¹⁷ Setdata4df32()	540
15.65.2.10 ¹⁸ Setdata4df64()	540
15.65.2.10 ¹⁹ Setdata4di16()	541
15.65.2.11 ¹⁹ Setdata4di32()	541
15.65.2.11 ²⁰ Setdata4di64()	542
15.65.2.11 ²¹ Setdata4di8()	542
15.65.2.11 ²² Setdata5df32()	543
15.65.2.11 ²³ Setdata5df64()	543
15.65.2.11 ²⁴ Setdata5di16()	544
15.65.2.11 ²⁵ Setdata5di32()	544
15.65.2.11 ²⁶ Setdata5di64()	545
15.65.2.11 ²⁷ Setdata5di8()	545
15.65.2.11 ²⁸ Setdatascalarf32()	546
15.65.2.12 ²⁹ Setdatascalarf64()	546
15.65.2.12 ³⁰ Setdatascalarf16()	547
15.65.2.12 ³¹ Setdatascalari32()	547
15.65.2.12 ³² Setdatascalari64()	547
15.65.2.12 ³³ Setdatascalari8()	548
15.65.2.12 ³⁴ Setdimension()	548
15.65.2.12 ³⁵ Setglobalattributechar()	549
15.65.2.12 ³⁶ Setglobalattributef32()	549
15.65.2.12 ³⁷ Setglobalattributef64()	549
15.65.2.12 ³⁸ Setglobalattributei16()	550
15.65.2.13 ³⁹ Setglobalattributei32()	550
15.65.2.13 ⁴⁰ Setglobalattributei64()	551
15.65.2.13 ⁴¹ Setglobalattributei8()	551
15.65.2.13 ⁴² Setvariableattributechar()	551
15.65.2.13 ⁴³ Setvariableattributef32()	552
15.65.2.13 ⁴⁴ Setvariableattributef64()	552
15.65.2.13 ⁴⁵ Setvariableattributei16()	553
15.65.2.13 ⁴⁶ Setvariableattributei32()	553
15.65.2.13 ⁴⁷ Setvariableattributei64()	553
15.65.2.13 ⁴⁸ Setvariableattributei8()	554
15.65.2.14 ⁴⁹ Setvariablefillvaluef32()	554
15.65.2.14 ⁵⁰ Setvariablefillvaluef64()	554
15.65.2.14 ⁵¹ Setvariablefillvaluei16()	554
15.65.2.14 ⁵² Setvariablefillvaluei32()	555
15.65.2.14 ⁵³ Setvariablefillvaluei64()	555
15.65.2.14 ⁵⁴ Setvariablefillvaluei8()	555

15.65.2.146 setvariablewithids()	555
15.65.2.147 setvariablewithnames()	556
15.65.2.148 setvariablewithtypes()	557
15.66 mo_neurons Module Reference	557
15.66.1 Detailed Description	558
15.66.2 Function/Subroutine Documentation	558
15.66.2.1 approx_mon_int()	558
15.66.2.2 approx_mon_int_eps()	559
15.66.2.3 approx_mon_int_steps()	560
15.66.2.4 cosmic()	561
15.66.2.5 desiletsn0()	562
15.66.2.6 intgrandfast()	563
15.66.2.7 lookupintegral()	564
15.66.2.8 oldintegration()	564
15.66.2.9 tabularintegralafast()	565
15.67 mo_nml Module Reference	566
15.67.1 Detailed Description	567
15.67.2 Function/Subroutine Documentation	567
15.67.2.1 close_nml()	567
15.67.2.2 open_nml()	568
15.67.2.3 position_nml()	569
15.67.3 Variable Documentation	571
15.67.3.1 length_error	571
15.67.3.2 missing	571
15.67.3.3 nunitnml	571
15.67.3.4 positioned	571
15.67.3.5 read_error	572
15.68 mo_objective_function Module Reference	572
15.68.1 Detailed Description	572
15.68.2 Function/Subroutine Documentation	573
15.68.2.1 extract_basin_avg_tws()	573
15.68.2.2 objective()	574
15.68.2.3 objective_et_kge_catchment_avg()	576
15.68.2.4 objective_kge_q_et()	577
15.68.2.5 objective_kge_q_rmse_et()	578
15.68.2.6 objective_kge_q_rmse_tws()	579
15.68.2.7 objective_kge_q_sm_corr()	581
15.68.2.8 objective_neutrons_kge_catchment_avg()	582
15.68.2.9 objective_sm_corr()	583
15.68.2.10 objective_sm_kge_catchment_avg()	585

15.68.2.11objective_sm_pd()	586
15.68.2.12objective_sm_sse_standard_score()	587
15.69mo_optimization Module Reference	588
15.69.1 Detailed Description	589
15.69.2 Function/Subroutine Documentation	589
15.69.2.1 optimization()	589
15.70mo_optimization_utils Module Reference	590
15.71mo_orderpack Module Reference	591
15.71.1 Detailed Description	593
15.71.2 Function/Subroutine Documentation	593
15.71.2.1 d_ctrper()	593
15.71.2.2 d_fndnth()	593
15.71.2.3 d_indmed()	593
15.71.2.4 d_indnth()	594
15.71.2.5 d_inspar()	594
15.71.2.6 d_inssor()	594
15.71.2.7 d_med()	594
15.71.2.8 d_median()	595
15.71.2.9 d_mrgref()	595
15.71.2.10d_mrgrnk()	595
15.71.2.11d_mulcnt()	595
15.71.2.12d_nearless()	595
15.71.2.13d_rapknr()	595
15.71.2.14d_refpar()	596
15.71.2.15d_refsor()	596
15.71.2.16d_rinpar()	596
15.71.2.17d_rnkpar()	596
15.71.2.18d_substor()	597
15.71.2.19d_uniinv()	597
15.71.2.20d_unipar()	597
15.71.2.21d_unirnk()	597
15.71.2.22d_unista()	597
15.71.2.23d_valmed()	598
15.71.2.24d_valnth()	598
15.71.2.25_ctrper()	598
15.71.2.26_fndnth()	598
15.71.2.27_indmed()	598
15.71.2.28_indnth()	598
15.71.2.29_inspar()	599
15.71.2.30_inssor()	599

15.71.2.31i_med()	599
15.71.2.32i_median()	600
15.71.2.33i_mrgref()	600
15.71.2.34i_mrgrnk()	600
15.71.2.35i_mulcnt()	600
15.71.2.36i_nearless()	600
15.71.2.37i_rapknr()	600
15.71.2.38i_refpar()	600
15.71.2.39i_refsor()	601
15.71.2.40i_rinpar()	601
15.71.2.41i_rnkpar()	601
15.71.2.42i_subssor()	601
15.71.2.43i_uniinv()	602
15.71.2.44i_unipar()	602
15.71.2.45i_unirnk()	602
15.71.2.46i_unista()	602
15.71.2.47i_valmed()	602
15.71.2.48i_valnth()	603
15.71.2.49i_ctrper()	603
15.71.2.50i_fndnth()	603
15.71.2.51r_indmed()	603
15.71.2.52r_indnth()	603
15.71.2.53r_insspar()	603
15.71.2.54r_inssor()	604
15.71.2.55r_med()	604
15.71.2.56r_median()	604
15.71.2.57r_mrgref()	605
15.71.2.58r_mrgrnk()	605
15.71.2.59r_mulcnt()	605
15.71.2.60r_nearless()	605
15.71.2.61r_rapknr()	605
15.71.2.62r_refpar()	605
15.71.2.63r_refsor()	605
15.71.2.64r_rinpar()	606
15.71.2.65r_rnkpar()	606
15.71.2.66r_subssor()	606
15.71.2.67r_uniinv()	607
15.71.2.68r_unipar()	607
15.71.2.69r_unirnk()	607
15.71.2.70r_unista()	607

15.71.2.71r_valmed()	607
15.71.2.72r_valnth()	607
15.71.2.73sort_index_dp()	607
15.71.2.74sort_index_i4()	608
15.71.2.75sort_index_sp()	608
15.71.3 Variable Documentation	608
15.71.3.1 idont	608
15.72mo_percentile Module Reference	608
15.72.1 Function/Subroutine Documentation	608
15.72.1.1 median_dp()	609
15.72.1.2 median_sp()	609
15.72.1.3 n_element_dp()	609
15.72.1.4 n_element_sp()	609
15.72.1.5 percentile_0d_dp()	609
15.72.1.6 percentile_0d_sp()	609
15.72.1.7 percentile_1d_dp()	610
15.72.1.8 percentile_1d_sp()	610
15.72.1.9 qmedian_dp()	610
15.72.1.10qmedian_sp()	610
15.73mo_pet Module Reference	610
15.73.1 Detailed Description	611
15.73.2 Function/Subroutine Documentation	611
15.73.2.1 extraterr_rad_approx()	611
15.73.2.2 pet_hargreaves()	612
15.73.2.3 pet_penman()	613
15.73.2.4 pet_priestly()	615
15.73.2.5 sat_vap_pressure()	616
15.73.2.6 slope_satpressure()	617
15.74mo_prepare_gridded_lai Module Reference	618
15.74.1 Detailed Description	618
15.74.2 Function/Subroutine Documentation	619
15.74.2.1 prepare_gridded_daily_lai_data()	619
15.74.2.2 prepare_gridded_mean_monthly_lai_data()	620
15.75mo_read_forcing_nc Module Reference	621
15.75.1 Detailed Description	621
15.75.2 Function/Subroutine Documentation	622
15.75.2.1 get_time_vector_and_select()	622
15.75.2.2 read_forcing_nc()	623
15.75.2.3 read_weights_nc()	625
15.76mo_read_lation Module Reference	626

15.76.1 Detailed Description	626
15.76.2 Function/Subroutine Documentation	627
15.76.2.1 <code>read_latlon()</code>	627
15.77 <code>mo_read_lut</code> Module Reference	628
15.77.1 Detailed Description	628
15.77.2 Function/Subroutine Documentation	629
15.77.2.1 <code>read_geoformation_lut()</code>	629
15.77.2.2 <code>read_lai_lut()</code>	630
15.78 <code>mo_read_optional_data</code> Module Reference	631
15.78.1 Detailed Description	631
15.78.2 Function/Subroutine Documentation	631
15.78.2.1 <code>read_basin_avg_tws()</code>	632
15.78.2.2 <code>read_evapotranspiration()</code>	632
15.78.2.3 <code>read_neutrons()</code>	633
15.78.2.4 <code>read_soil_moisture()</code>	634
15.79 <code>mo_read_spatial_data</code> Module Reference	635
15.79.1 Detailed Description	636
15.79.2 Function/Subroutine Documentation	636
15.79.2.1 <code>read_header_ascii()</code>	636
15.79.2.2 <code>read_spatial_data_ascii_dp()</code>	637
15.79.2.3 <code>read_spatial_data_ascii_i4()</code>	638
15.80 <code>mo_read_timeseries</code> Module Reference	639
15.80.1 Detailed Description	639
15.80.2 Function/Subroutine Documentation	639
15.80.2.1 <code>read_timeseries()</code>	640
15.81 <code>mo_read_wrapper</code> Module Reference	641
15.81.1 Detailed Description	641
15.81.2 Function/Subroutine Documentation	642
15.81.2.1 <code>check_consistency_lut_map()</code>	642
15.81.2.2 <code>read_data()</code>	643
15.82 <code>mo_restart</code> Module Reference	644
15.82.1 Detailed Description	645
15.82.2 Function/Subroutine Documentation	645
15.82.2.1 <code>read_restart_states()</code>	645
15.82.2.2 <code>unpack_field_and_write_1d_dp()</code>	646
15.82.2.3 <code>unpack_field_and_write_1d_i4()</code>	646
15.82.2.4 <code>unpack_field_and_write_2d_dp()</code>	647
15.82.2.5 <code>unpack_field_and_write_3d_dp()</code>	647
15.82.2.6 <code>write_restart_files()</code>	647
15.83 <code>mo_runoff</code> Module Reference	648

15.83.1 Detailed Description	649
15.83.2 Function/Subroutine Documentation	649
15.83.2.1 l1_total_runoff()	649
15.83.2.2 runoff_sat_zone()	650
15.83.2.3 runoff_unsat_zone()	651
15.84 mo_sce Module Reference	652
15.84.1 Detailed Description	652
15.84.2 Function/Subroutine Documentation	652
15.84.2.1 cce()	652
15.84.2.2 chkcst()	653
15.84.2.3 comp()	654
15.84.2.4 getpnt()	654
15.84.2.5 parsrt()	655
15.84.2.6 sce()	655
15.84.2.7 sort_matrix()	659
15.85 mo_set_netcdf_outputs Module Reference	660
15.85.1 Detailed Description	660
15.85.2 Function/Subroutine Documentation	660
15.85.2.1 set_netcdf()	660
15.86 mo_snow_accum_melt Module Reference	661
15.86.1 Detailed Description	661
15.86.2 Function/Subroutine Documentation	661
15.86.2.1 snow_accum_melt()	661
15.87 mo_soil_database Module Reference	662
15.87.1 Detailed Description	663
15.87.2 Function/Subroutine Documentation	663
15.87.2.1 generate_soil_database()	663
15.87.2.2 read_soil_lut()	664
15.88 mo_soil_moisture Module Reference	665
15.88.1 Detailed Description	665
15.88.2 Function/Subroutine Documentation	665
15.88.2.1 feddes_et_reduction()	666
15.88.2.2 jarvis_et_reduction()	666
15.88.2.3 soil_moisture()	667
15.89 mo_spatial_agg_disagg_forcing Module Reference	669
15.89.1 Detailed Description	670
15.89.2 Function/Subroutine Documentation	670
15.89.2.1 spatial_aggregation_3d()	670
15.89.2.2 spatial_aggregation_4d()	670
15.89.2.3 spatial_disaggregation_3d()	671

15.89.2.4 spatial_disaggregation_4d()	671
15.90 mo_spatialsimilarity Module Reference	671
15.90.1 Detailed Description	671
15.90.2 Function/Subroutine Documentation	672
15.90.2.1 nndv_dp()	672
15.90.2.2 nndv_sp()	672
15.90.2.3 pd_dp()	672
15.90.2.4 pd_sp()	672
15.91 mo_standard_score Module Reference	672
15.91.1 Detailed Description	673
15.91.2 Function/Subroutine Documentation	673
15.91.2.1 classified_standard_score_dp()	673
15.91.2.2 classified_standard_score_sp()	673
15.91.2.3 standard_score_dp()	674
15.91.2.4 standard_score_sp()	674
15.92 mo_startup Module Reference	674
15.92.1 Detailed Description	674
15.92.2 Function/Subroutine Documentation	674
15.92.2.1 constants_init()	674
15.92.2.2 l2_variable_init()	675
15.92.2.3 mhm_initialize()	676
15.93 mo_string_utils Module Reference	678
15.93.1 Detailed Description	678
15.93.2 Function/Subroutine Documentation	678
15.93.2.1 compress()	679
15.93.2.2 divide_string()	679
15.93.2.3 dp2str()	680
15.93.2.4 equalstrings()	680
15.93.2.5 i42str()	680
15.93.2.6 i4array2str()	681
15.93.2.7 i82str()	681
15.93.2.8 log2str()	681
15.93.2.9 nonull()	681
15.93.2.10sp2str()	682
15.93.2.11splitstring()	682
15.93.2.12startswith()	682
15.93.2.13str2num()	683
15.93.2.14tolower()	683
15.93.2.15toupper()	684
15.93.3 Variable Documentation	684

15.93.3.1 <code>separator</code>	685
15.94 <code>mo_template</code> Module Reference	685
15.94.1 Detailed Description	685
15.94.2 Function/Subroutine Documentation	685
15.94.2.1 <code>circum()</code>	686
15.94.2.2 <code>mean_dp()</code>	686
15.94.2.3 <code>mean_sp()</code>	686
15.94.3 Variable Documentation	686
15.94.3.1 <code>itest</code>	686
15.94.3.2 <code>pi_dp</code>	687
15.94.3.3 <code>pi_sp</code>	687
15.95 <code>mo_temporal_aggregation</code> Module Reference	687
15.95.1 Detailed Description	687
15.95.2 Function/Subroutine Documentation	687
15.95.2.1 <code>day2mon_average_dp()</code>	688
15.95.2.2 <code>hour2day_average_dp()</code>	688
15.96 <code>mo_temporal_disagg_forcing</code> Module Reference	688
15.96.1 Detailed Description	689
15.96.2 Function/Subroutine Documentation	689
15.96.2.1 <code>temporal_disagg_forcing()</code>	689
15.97 <code>mo_timer</code> Module Reference	690
15.97.1 Detailed Description	691
15.97.2 Function/Subroutine Documentation	691
15.97.2.1 <code>timer_check()</code>	691
15.97.2.2 <code>timer_clear()</code>	692
15.97.2.3 <code>timer_get()</code>	693
15.97.2.4 <code>timer_print()</code>	694
15.97.2.5 <code>timer_start()</code>	695
15.97.2.6 <code>timer_stop()</code>	696
15.97.2.7 <code>timers_init()</code>	697
15.97.3 Variable Documentation	698
15.97.3.1 <code>clock_rate</code>	698
15.97.3.2 <code>cputime</code>	698
15.97.3.3 <code>cycles1</code>	698
15.97.3.4 <code>cycles2</code>	698
15.97.3.5 <code>cycles_max</code>	699
15.97.3.6 <code>max_timers</code>	699
15.97.3.7 <code>status</code>	699
15.98 <code>mo_upscaling_operators</code> Module Reference	699
15.98.1 Detailed Description	700

15.98.2 Function/Subroutine Documentation	700
15.98.2.1 l0_fractionalcover_in_lx()	700
15.98.2.2 majority_statistics()	701
15.98.2.3 upscale_arithmetic_mean()	702
15.98.2.4 upscale_geometric_mean()	703
15.98.2.5 upscale_harmonic_mean()	704
15.98.2.6 upscale_p_norm()	705
15.99 mo_utils Module Reference	706
15.99.1 Detailed Description	707
15.99.2 Function/Subroutine Documentation	707
15.99.2.1 equal_dp()	707
15.99.2.2 equal_sp()	707
15.99.2.3 greaterequal_dp()	707
15.99.2.4 greaterequal_sp()	708
15.99.2.5 is_finite_dp()	708
15.99.2.6 is_finite_sp()	708
15.99.2.7 is_nan_dp()	708
15.99.2.8 is_nan_sp()	708
15.99.2.9 is_normal_dp()	708
15.99.2.10 is_normal_sp()	708
15.99.2.11 lesserequal_dp()	708
15.99.2.12 lesserequal_sp()	709
15.99.2.13 locate_0d_dp()	709
15.99.2.14 locate_0d_sp()	709
15.99.2.15 locate_1d_dp()	709
15.99.2.16 locate_1d_sp()	709
15.99.2.17 notequal_dp()	709
15.99.2.18 notequal_sp()	709
15.99.2.19 special_value_dp()	710
15.99.2.20 special_value_sp()	710
15.99.2.21 swap_vec_dp()	710
15.99.2.22 swap_vec_i4()	710
15.99.2.23 swap_vec_sp()	711
15.99.2.24 swap_xy_dp()	711
15.99.2.25 swap_xy_i4()	711
15.99.2.26 swap_xy_sp()	711
15.10 mo_write_ascii Module Reference	711
15.100.1 Detailed Description	711
15.100.2 Function/Subroutine Documentation	712
15.100.2.1 write_configfile()	712

15.100.2.2	write_optfile()	713
15.100.2.3	write_optinamelist()	714
15.101	no_write_fluxes_states Module Reference	715
15.101.1	Detailed Description	716
15.101.2	Function/Subroutine Documentation	716
15.101.2.1	close()	716
15.101.2.2	createoutputfile()	716
15.101.2.3	fluxesunit()	717
15.101.2.4	newoutputdataset()	718
15.101.2.5	newoutputvariable()	719
15.101.2.6	updatedataset()	720
15.101.2.7	updatevariable()	722
15.101.2.8	writetimestep()	722
15.101.2.9	writevariableattributes()	723
15.101.2.10	writevariabletimestep()	724
15.102	no_xor4096 Module Reference	724
15.102.1	Function/Subroutine Documentation	725
15.102.1.1	get_timeseed_i4_0d()	725
15.102.1.2	get_timeseed_i4_1d()	725
15.102.1.3	get_timeseed_i8_0d()	725
15.102.1.4	get_timeseed_i8_1d()	725
15.102.1.5	xor4096d_0d()	725
15.102.1.6	xor4096d_1d()	725
15.102.1.7	xor4096f_0d()	726
15.102.1.8	xor4096f_1d()	726
15.102.1.9	xor4096gd_0d()	726
15.102.1.10	xor4096gd_1d()	726
15.102.1.11	xor4096gf_0d()	726
15.102.1.12	xor4096gf_1d()	727
15.102.1.13	xor4096l_0d()	727
15.102.1.14	xor4096l_1d()	727
15.102.1.15	xor4096s_0d()	727
15.102.1.16	xor4096s_1d()	727
15.102.2	Variable Documentation	728
15.102.2.1	ln_save_state	728
16	Data Type Documentation	729
16.1	mo_moment::absdev Interface Reference	729
16.1.1	Member Function/Subroutine Documentation	729
16.1.1.1	absdev_dp()	729

16.1.1.2 <code>absdev_sp()</code>	729
16.2 <code>mo_anneal::anneal</code> Interface Reference	729
16.2.1 Detailed Description	730
16.2.2 Member Function/Subroutine Documentation	731
16.2.2.1 <code>anneal_dp()</code>	731
16.3 <code>mo_append::append</code> Interface Reference	732
16.3.1 Detailed Description	732
16.3.2 Member Function/Subroutine Documentation	734
16.3.2.1 <code>append_char_3d()</code>	734
16.3.2.2 <code>append_char_m_m()</code>	734
16.3.2.3 <code>append_char_v_s()</code>	734
16.3.2.4 <code>append_char_v_v()</code>	734
16.3.2.5 <code>append_dp_3d()</code>	735
16.3.2.6 <code>append_dp_m_m()</code>	735
16.3.2.7 <code>append_dp_v_s()</code>	735
16.3.2.8 <code>append_dp_v_v()</code>	735
16.3.2.9 <code>append_i4_m_m()</code>	735
16.3.2.10 <code>append_i4_v_s()</code>	735
16.3.2.11 <code>append_i4_v_v()</code>	735
16.3.2.12 <code>append_i8_3d()</code>	736
16.3.2.13 <code>append_i8_m_m()</code>	736
16.3.2.14 <code>append_i8_v_s()</code>	736
16.3.2.15 <code>append_i8_v_v()</code>	736
16.3.2.16 <code>append_lgt_3d()</code>	736
16.3.2.17 <code>append_lgt_m_m()</code>	736
16.3.2.18 <code>append_lgt_v_s()</code>	737
16.3.2.19 <code>append_lgt_v_v()</code>	737
16.3.2.20 <code>append_sp_3d()</code>	737
16.3.2.21 <code>append_sp_m_m()</code>	737
16.3.2.22 <code>append_sp_v_s()</code>	737
16.3.2.23 <code>append_sp_v_v()</code>	737
16.4 <code>mo_corr::arth</code> Interface Reference	737
16.4.1 Member Function/Subroutine Documentation	738
16.4.1.1 <code>arth_dp()</code>	738
16.4.1.2 <code>arth_i4()</code>	738
16.4.1.3 <code>arth_sp()</code>	738
16.5 <code>mo_ncwrite::attribute</code> Type Reference	739
16.5.1 Member Data Documentation	739
16.5.1.1 <code>name</code>	739
16.5.1.2 <code>nvalues</code>	739

16.5.1.3	values	739
16.5.1.4	xtype	739
16.6	mo_corr::autocoeffk Interface Reference	740
16.6.1	Member Function/Subroutine Documentation	740
16.6.1.1	autocoeffk_1d_dp()	740
16.6.1.2	autocoeffk_1d_sp()	740
16.6.1.3	autocoeffk_dp()	740
16.6.1.4	autocoeffk_sp()	740
16.7	mo_corr::autocorr Interface Reference	741
16.7.1	Member Function/Subroutine Documentation	741
16.7.1.1	autocorr_1d_dp()	741
16.7.1.2	autocorr_1d_sp()	741
16.7.1.3	autocorr_dp()	741
16.7.1.4	autocorr_sp()	741
16.8	mo_moment::average Interface Reference	741
16.8.1	Member Function/Subroutine Documentation	742
16.8.1.1	average_dp()	742
16.8.1.2	average_sp()	742
16.9	mo_mrm_global_variables::basininfo_mrm Type Reference	742
16.9.1	Member Data Documentation	743
16.9.1.1	gaugeidlist	743
16.9.1.2	gaugeindexlist	743
16.9.1.3	gaugenodelist	743
16.9.1.4	inflowgaugeheadwater	743
16.9.1.5	inflowgaugeidlist	743
16.9.1.6	inflowgaugeindexlist	743
16.9.1.7	inflowgaugenodelist	744
16.9.1.8	l0_coloutlet	744
16.9.1.9	l0_noutlet	744
16.9.1.10	l0_rowoutlet	744
16.9.1.11	ngauges	744
16.9.1.12	ninflowgauges	744
16.10	mo_errormeasures::bias Interface Reference	744
16.10.1	Member Function/Subroutine Documentation	744
16.10.1.1	bias_dp_1d()	745
16.10.1.2	bias_dp_2d()	745
16.10.1.3	bias_dp_3d()	745
16.10.1.4	bias_sp_1d()	745
16.10.1.5	bias_sp_2d()	745
16.10.1.6	bias_sp_3d()	745

16.11	mo_moment::central_moment Interface Reference	746
16.11.1	Member Function/Subroutine Documentation	746
16.11.1.1	central_moment_dp()	746
16.11.1.2	central_moment_sp()	746
16.12	mo_moment::central_moment_var Interface Reference	746
16.12.1	Member Function/Subroutine Documentation	746
16.12.1.1	central_moment_var_dp()	746
16.12.1.2	central_moment_var_sp()	747
16.13	mo_standard_score::classified_standard_score Interface Reference	747
16.13.1	Detailed Description	747
16.13.2	Member Function/Subroutine Documentation	748
16.13.2.1	classified_standard_score_dp()	748
16.13.2.2	classified_standard_score_sp()	748
16.14	mo_corr::corr Interface Reference	748
16.14.1	Member Function/Subroutine Documentation	748
16.14.1.1	corr_dp()	748
16.14.1.2	corr_sp()	749
16.15	mo_moment::correlation Interface Reference	749
16.15.1	Member Function/Subroutine Documentation	749
16.15.1.1	correlation_dp()	749
16.15.1.2	correlation_sp()	749
16.16	mo_moment::covariance Interface Reference	749
16.16.1	Member Function/Subroutine Documentation	750
16.16.1.1	covariance_dp()	750
16.16.1.2	covariance_sp()	750
16.17	mo_corr::crosscoeffk Interface Reference	750
16.17.1	Member Function/Subroutine Documentation	750
16.17.1.1	crosscoeffk_dp()	750
16.17.1.2	crosscoeffk_sp()	751
16.18	mo_corr::crosscorr Interface Reference	751
16.18.1	Member Function/Subroutine Documentation	751
16.18.1.1	crosscorr_dp()	751
16.18.1.2	crosscorr_sp()	751
16.19	mo_orderpack::ctrper Interface Reference	751
16.19.1	Member Function/Subroutine Documentation	752
16.19.1.1	d_ctrper()	752
16.19.1.2	i_ctrper()	752
16.19.1.3	r_ctrper()	752
16.20	mo_temporal_aggregation::day2mon_average Interface Reference	752
16.20.1	Detailed Description	752

16.20.2 Member Function/Subroutine Documentation	753
16.20.2.1 day2mon_average_dp()	753
16.21 mo_ncwrite::dims Type Reference	753
16.21.1 Member Data Documentation	754
16.21.1.1 dimid	754
16.21.1.2 len	754
16.21.1.3 name	754
16.22 mo_ncwrite::dump_netcdf Interface Reference	754
16.22.1 Member Function/Subroutine Documentation	754
16.22.1.1 dump_netcdf_1d_dp()	755
16.22.1.2 dump_netcdf_1d_i4()	755
16.22.1.3 dump_netcdf_1d_sp()	755
16.22.1.4 dump_netcdf_2d_dp()	755
16.22.1.5 dump_netcdf_2d_i4()	755
16.22.1.6 dump_netcdf_2d_sp()	756
16.22.1.7 dump_netcdf_3d_dp()	756
16.22.1.8 dump_netcdf_3d_i4()	756
16.22.1.9 dump_netcdf_3d_sp()	756
16.22.1.10 dump_netcdf_4d_dp()	756
16.22.1.11 dump_netcdf_4d_i4()	757
16.22.1.12 dump_netcdf_4d_sp()	757
16.22.1.13 dump_netcdf_5d_dp()	757
16.22.1.14 dump_netcdf_5d_i4()	757
16.22.1.15 dump_netcdf_5d_sp()	757
16.23 mo_utils::eq Interface Reference	758
16.23.1 Member Function/Subroutine Documentation	758
16.23.1.1 equal_dp()	758
16.23.1.2 equal_sp()	758
16.24 mo_utils::equal Interface Reference	758
16.24.1 Detailed Description	759
16.24.2 Member Function/Subroutine Documentation	759
16.24.2.1 equal_dp()	759
16.24.2.2 equal_sp()	759
16.25 mo_optimization_utils::eval_interface Interface Reference	759
16.25.1 Constructor & Destructor Documentation	760
16.25.1.1 eval_interface()	760
16.26 mo_orderpack::fndnth Interface Reference	760
16.26.1 Member Function/Subroutine Documentation	760
16.26.1.1 d_fndnth()	760
16.26.1.2 i_fndnth()	760

16.26.1.3 r_fndnth()	760
16.27 mo_corr::four1 Interface Reference	761
16.27.1 Member Function/Subroutine Documentation	761
16.27.1.1 four1_dp()	761
16.27.1.2 four1_sp()	761
16.28 mo_corr::fourrow Interface Reference	761
16.28.1 Member Function/Subroutine Documentation	761
16.28.1.1 fourrow_dp()	761
16.28.1.2 fourrow_sp()	762
16.29 mo_mrm_global_variables::gaugingstation Type Reference	762
16.29.1 Member Data Documentation	762
16.29.1.1 basinid	762
16.29.1.2 fname	762
16.29.1.3 gaugeid	763
16.29.1.4 q	763
16.30 mo_utils::ge Interface Reference	763
16.30.1 Member Function/Subroutine Documentation	763
16.30.1.1 greaterequal_dp()	763
16.30.1.2 greaterequal_sp()	763
16.31 mo_anneal::generate_neighborhood_weight Interface Reference	763
16.31.1 Member Function/Subroutine Documentation	763
16.31.1.1 generate_neighborhood_weight_dp()	764
16.32 mo_ncread::get_ncvar Interface Reference	764
16.32.1 Member Function/Subroutine Documentation	764
16.32.1.1 get_ncvar_0d_dp()	764
16.32.1.2 get_ncvar_0d_i1()	765
16.32.1.3 get_ncvar_0d_i4()	765
16.32.1.4 get_ncvar_0d_sp()	765
16.32.1.5 get_ncvar_1d_dp()	765
16.32.1.6 get_ncvar_1d_i1()	765
16.32.1.7 get_ncvar_1d_i4()	765
16.32.1.8 get_ncvar_1d_sp()	766
16.32.1.9 get_ncvar_2d_dp()	766
16.32.1.10 get_ncvar_2d_i1()	766
16.32.1.11 get_ncvar_2d_i4()	766
16.32.1.12 get_ncvar_2d_sp()	767
16.32.1.13 get_ncvar_3d_dp()	767
16.32.1.14 get_ncvar_3d_i1()	767
16.32.1.15 get_ncvar_3d_i4()	767
16.32.1.16 get_ncvar_3d_sp()	767

16.32.1.17get_ncvar_4d_dp()	768
16.32.1.18get_ncvar_4d_i1()	768
16.32.1.19get_ncvar_4d_i4()	768
16.32.1.20get_ncvar_4d_sp()	768
16.32.1.21get_ncvar_5d_dp()	768
16.32.1.22get_ncvar_5d_i1()	769
16.32.1.23get_ncvar_5d_i4()	769
16.32.1.24get_ncvar_5d_sp()	769
16.33mo_xor4096::get_timeseed Interface Reference	769
16.33.1 Member Function/Subroutine Documentation	769
16.33.1.1 get_timeseed_i4_0d()	770
16.33.1.2 get_timeseed_i4_1d()	770
16.33.1.3 get_timeseed_i8_0d()	770
16.33.1.4 get_timeseed_i8_1d()	770
16.34mo_anneal::gettemperature Interface Reference	770
16.34.1 Detailed Description	770
16.34.2 Member Function/Subroutine Documentation	771
16.34.2.1 gettemperature_dp()	771
16.35mo_utils::greaterequal Interface Reference	772
16.35.1 Member Function/Subroutine Documentation	772
16.35.1.1 greaterequal_dp()	772
16.35.1.2 greaterequal_sp()	772
16.36mo_common_variables::grid Type Reference	773
16.36.1 Member Data Documentation	773
16.36.1.1 cellarea	773
16.36.1.2 cellcoor	774
16.36.1.3 cellsize	774
16.36.1.4 id	774
16.36.1.5 iend	774
16.36.1.6 istart	774
16.36.1.7 mask	774
16.36.1.8 ncells	774
16.36.1.9 ncols	774
16.36.1.10nodata_value	774
16.36.1.11nrows	775
16.36.1.12xllcorner	775
16.36.1.13yllcorner	775
16.36.1.14y	775
16.36.1.15yllcorner	775
16.37mo_common_variables::gridremapper Type Reference	776

16.37.1 Member Data Documentation	776
16.37.1.1 high_res_grid	776
16.37.1.2 left_bound	777
16.37.1.3 low_res_grid	777
16.37.1.4 lower_bound	777
16.37.1.5 lowres_id_on_highres	777
16.37.1.6 n_subcells	777
16.37.1.7 right_bound	777
16.37.1.8 upper_bound	777
16.38 mo_temporal_aggregation::hour2day_average Interface Reference	777
16.38.1 Detailed Description	778
16.38.2 Member Function/Subroutine Documentation	778
16.38.2.1 hour2day_average_dp()	778
16.39 mo_orderpack::indmed Interface Reference	778
16.39.1 Member Function/Subroutine Documentation	779
16.39.1.1 d_indmed()	779
16.39.1.2 i_indmed()	779
16.39.1.3 r_indmed()	779
16.40 mo_orderpack::indnth Interface Reference	779
16.40.1 Member Function/Subroutine Documentation	779
16.40.1.1 d_indnth()	779
16.40.1.2 i_indnth()	780
16.40.1.3 r_indnth()	780
16.41 mo_orderpack::inspar Interface Reference	780
16.41.1 Member Function/Subroutine Documentation	780
16.41.1.1 d_inspars()	780
16.41.1.2 i_inspars()	780
16.41.1.3 r_inspars()	780
16.42 mo_orderpack::inssor Interface Reference	781
16.42.1 Member Function/Subroutine Documentation	781
16.42.1.1 d_inssor()	781
16.42.1.2 i_inssor()	781
16.42.1.3 r_inssor()	781
16.43 mo_utils::is_finite Interface Reference	781
16.43.1 Detailed Description	781
16.43.2 Member Function/Subroutine Documentation	782
16.43.2.1 is_finite_dp()	782
16.43.2.2 is_finite_sp()	782
16.44 mo_utils::is_nan Interface Reference	782
16.44.1 Member Function/Subroutine Documentation	782

16.44.1.1 <code>is_nan_dp()</code>	782
16.44.1.2 <code>is_nan_sp()</code>	783
16.45 <code>mo_utils::is_normal</code> Interface Reference	783
16.45.1 Member Function/Subroutine Documentation	783
16.45.1.1 <code>is_normal_dp()</code>	783
16.45.1.2 <code>is_normal_sp()</code>	783
16.46 <code>mo_errormeasures::kge</code> Interface Reference	783
16.46.1 Detailed Description	784
16.46.2 Member Function/Subroutine Documentation	784
16.46.2.1 <code>kge_dp_1d()</code>	784
16.46.2.2 <code>kge_dp_2d()</code>	785
16.46.2.3 <code>kge_dp_3d()</code>	785
16.46.2.4 <code>kge_sp_1d()</code>	785
16.46.2.5 <code>kge_sp_2d()</code>	785
16.46.2.6 <code>kge_sp_3d()</code>	785
16.47 <code>mo_errormeasures::kgenocorr</code> Interface Reference	785
16.47.1 Detailed Description	786
16.47.2 Member Function/Subroutine Documentation	786
16.47.2.1 <code>kgenocorr_dp_1d()</code>	786
16.47.2.2 <code>kgenocorr_dp_2d()</code>	787
16.47.2.3 <code>kgenocorr_dp_3d()</code>	787
16.47.2.4 <code>kgenocorr_sp_1d()</code>	787
16.47.2.5 <code>kgenocorr_sp_2d()</code>	787
16.47.2.6 <code>kgenocorr_sp_3d()</code>	787
16.48 <code>mo_moment::kurtosis</code> Interface Reference	787
16.48.1 Member Function/Subroutine Documentation	788
16.48.1.1 <code>kurtosis_dp()</code>	788
16.48.1.2 <code>kurtosis_sp()</code>	788
16.49 <code>mo_utils::le</code> Interface Reference	788
16.49.1 Member Function/Subroutine Documentation	788
16.49.1.1 <code>lesserequal_dp()</code>	788
16.49.1.2 <code>lesserequal_sp()</code>	788
16.50 <code>mo_utils::lesserequal</code> Interface Reference	789
16.50.1 Member Function/Subroutine Documentation	789
16.50.1.1 <code>lesserequal_dp()</code>	789
16.50.1.2 <code>lesserequal_sp()</code>	789
16.51 <code>mo_linfit::linfit</code> Interface Reference	789
16.51.1 Detailed Description	789
16.51.2 Member Function/Subroutine Documentation	790
16.51.2.1 <code>linfit_dp()</code>	790

16.51.2.2 linfit_sp()	790
16.52mo_errormeasures::lnnse Interface Reference	790
16.52.1 Member Function/Subroutine Documentation	791
16.52.1.1 lnnse_dp_1d()	791
16.52.1.2 lnnse_dp_2d()	791
16.52.1.3 lnnse_dp_3d()	791
16.52.1.4 lnnse_sp_1d()	791
16.52.1.5 lnnse_sp_2d()	791
16.52.1.6 lnnse_sp_3d()	791
16.53mo_utils::locate Interface Reference	792
16.53.1 Detailed Description	792
16.53.2 Member Function/Subroutine Documentation	792
16.53.2.1 locate_0d_dp()	792
16.53.2.2 locate_0d_sp()	793
16.53.2.3 locate_1d_dp()	793
16.53.2.4 locate_1d_sp()	793
16.54mo_errormeasures::mae Interface Reference	793
16.54.1 Member Function/Subroutine Documentation	793
16.54.1.1 mae_dp_1d()	793
16.54.1.2 mae_dp_2d()	793
16.54.1.3 mae_dp_3d()	794
16.54.1.4 mae_sp_1d()	794
16.54.1.5 mae_sp_2d()	794
16.54.1.6 mae_sp_3d()	794
16.55mo_mcmc::mcmc Interface Reference	794
16.55.1 Detailed Description	795
16.55.2 Member Function/Subroutine Documentation	798
16.55.2.1 mcmc_dp()	798
16.56mo_mcmc::mcmc_stddev Interface Reference	798
16.56.1 Detailed Description	799
16.56.2 Member Function/Subroutine Documentation	802
16.56.2.1 mcmc_stddev_dp()	802
16.57mo_template::mean Interface Reference	802
16.57.1 Detailed Description	803
16.57.2 Member Function/Subroutine Documentation	803
16.57.2.1 mean_dp()	803
16.57.2.2 mean_sp()	803
16.58mo_moment::mean Interface Reference	804
16.58.1 Member Function/Subroutine Documentation	804
16.58.1.1 mean_dp()	804

16.58.1.2 <code>mean_sp()</code>	804
16.59 <code>mo_percentile::median</code> Interface Reference	804
16.59.1 Member Function/Subroutine Documentation	804
16.59.1.1 <code>median_dp()</code>	804
16.59.1.2 <code>median_sp()</code>	804
16.60 <code>mo_moment::mixed_central_moment</code> Interface Reference	805
16.60.1 Member Function/Subroutine Documentation	805
16.60.1.1 <code>mixed_central_moment_dp()</code>	805
16.60.1.2 <code>mixed_central_moment_sp()</code>	805
16.61 <code>mo_moment::mixed_central_moment_var</code> Interface Reference	805
16.61.1 Member Function/Subroutine Documentation	805
16.61.1.1 <code>mixed_central_moment_var_dp()</code>	806
16.61.1.2 <code>mixed_central_moment_var_sp()</code>	806
16.62 <code>mo_moment::moment</code> Interface Reference	806
16.62.1 Member Function/Subroutine Documentation	806
16.62.1.1 <code>moment_dp()</code>	806
16.62.1.2 <code>moment_sp()</code>	806
16.63 <code>mo_orderpack::mrgref</code> Interface Reference	807
16.63.1 Member Function/Subroutine Documentation	807
16.63.1.1 <code>d_mrgref()</code>	807
16.63.1.2 <code>i_mrgref()</code>	807
16.63.1.3 <code>r_mrgref()</code>	807
16.64 <code>mo_orderpack::mrgrnk</code> Interface Reference	807
16.64.1 Member Function/Subroutine Documentation	808
16.64.1.1 <code>d_mrgrnk()</code>	808
16.64.1.2 <code>i_mrgrnk()</code>	808
16.64.1.3 <code>r_mrgrnk()</code>	808
16.65 <code>mo_errormeasures::mse</code> Interface Reference	808
16.65.1 Member Function/Subroutine Documentation	808
16.65.1.1 <code>mse_dp_1d()</code>	809
16.65.1.2 <code>mse_dp_2d()</code>	809
16.65.1.3 <code>mse_dp_3d()</code>	809
16.65.1.4 <code>mse_sp_1d()</code>	809
16.65.1.5 <code>mse_sp_2d()</code>	809
16.65.1.6 <code>mse_sp_3d()</code>	809
16.66 <code>mo_orderpack::mulcnt</code> Interface Reference	810
16.66.1 Member Function/Subroutine Documentation	810
16.66.1.1 <code>d_mulcnt()</code>	810
16.66.1.2 <code>i_mulcnt()</code>	810
16.66.1.3 <code>r_mulcnt()</code>	810

16.67 <code>mo_percentile::n_element</code> Interface Reference	810
16.67.1 Member Function/Subroutine Documentation	810
16.67.1.1 <code>n_element_dp()</code>	811
16.67.1.2 <code>n_element_sp()</code>	811
16.68 <code>mo_ncdf::ncdataset</code> Interface Reference	811
16.68.1 Detailed Description	813
16.68.2 Member Function/Subroutine Documentation	813
16.68.2.1 <code>close()</code>	813
16.68.2.2 <code>getattribute()</code>	814
16.68.2.3 <code>getdimension()</code>	814
16.68.2.4 <code>getdimensionbyid()</code>	814
16.68.2.5 <code>getdimensionbyname()</code>	815
16.68.2.6 <code>getglobalattributechar()</code>	815
16.68.2.7 <code>getglobalattributef32()</code>	815
16.68.2.8 <code>getglobalattributef64()</code>	815
16.68.2.9 <code>getglobalattributei16()</code>	815
16.68.2.10 <code>getglobalattributei32()</code>	815
16.68.2.11 <code>getglobalattributei64()</code>	815
16.68.2.12 <code>getglobalattributei8()</code>	815
16.68.2.13 <code>getnovariables()</code>	815
16.68.2.14 <code>getunlimiteddimension()</code>	816
16.68.2.15 <code>getvariable()</code>	816
16.68.2.16 <code>getvariablebyname()</code>	816
16.68.2.17 <code>getvariableids()</code>	817
16.68.2.18 <code>getvariables()</code>	817
16.68.2.19 <code>hasdimension()</code>	817
16.68.2.20 <code>hasvariable()</code>	817
16.68.2.21 <code>initncdataset()</code>	818
16.68.2.22 <code>sunlimited()</code>	818
16.68.2.23 <code>setattribute()</code>	818
16.68.2.24 <code>setdimension()</code>	819
16.68.2.25 <code>setglobalattributechar()</code>	819
16.68.2.26 <code>setglobalattributef32()</code>	819
16.68.2.27 <code>setglobalattributef64()</code>	819
16.68.2.28 <code>setglobalattributei16()</code>	819
16.68.2.29 <code>setglobalattributei32()</code>	820
16.68.2.30 <code>setglobalattributei64()</code>	820
16.68.2.31 <code>setglobalattributei8()</code>	820
16.68.2.32 <code>setvariable()</code>	820
16.68.2.33 <code>setvariablewithids()</code>	821

16.68.2.34setvariablewithnames()	821
16.68.2.35setvariablewithtypes()	821
16.68.3 Member Data Documentation	821
16.68.3.1 dataset	821
16.68.3.2 file	821
16.68.3.3 filename	821
16.68.3.4 fname	821
16.68.3.5 id	821
16.68.3.6 mode	822
16.68.3.7 netcdf	822
16.68.3.8 of	822
16.68.3.9 open	822
16.68.3.10opened	822
16.68.3.11the	822
16.69mo_ncdf::ncdimension Type Reference	822
16.69.1 Detailed Description	826
16.69.2 Member Function/Subroutine Documentation	826
16.69.2.1 getattribute()	826
16.69.2.2 getdata()	827
16.69.2.3 getdata1df32()	827
16.69.2.4 getdata1df64()	827
16.69.2.5 getdata1di16()	827
16.69.2.6 getdata1di32()	827
16.69.2.7 getdata1di64()	828
16.69.2.8 getdata1di8()	828
16.69.2.9 getdata2df32()	828
16.69.2.10getdata2df64()	828
16.69.2.11getdata2di16()	828
16.69.2.12getdata2di32()	828
16.69.2.13getdata2di64()	828
16.69.2.14getdata2di8()	828
16.69.2.15getdata3df32()	828
16.69.2.16getdata3df64()	829
16.69.2.17getdata3di16()	829
16.69.2.18getdata3di32()	829
16.69.2.19getdata3di64()	829
16.69.2.20getdata3di8()	829
16.69.2.21getdata4df32()	829
16.69.2.22getdata4df64()	829
16.69.2.23getdata4di16()	829

16.69.2.24getdata4di32()	829
16.69.2.25getdata4di64()	830
16.69.2.26getdata4di8()	830
16.69.2.27getdata5df32()	830
16.69.2.28getdata5df64()	830
16.69.2.29getdata5di16()	830
16.69.2.30getdata5di32()	830
16.69.2.31getdata5di64()	830
16.69.2.32getdata5di8()	830
16.69.2.33getdatascalarf32()	830
16.69.2.34getdatascalarf64()	831
16.69.2.35getdatascalari16()	831
16.69.2.36getdatascalari32()	831
16.69.2.37getdatascalari64()	831
16.69.2.38getdatascalari8()	831
16.69.2.39getdimensions()	831
16.69.2.40getdtype()	831
16.69.2.41getfillvalue()	831
16.69.2.42getname()	832
16.69.2.43getnodimensions()	832
16.69.2.44getshape()	832
16.69.2.45getvariableattributechar()	832
16.69.2.46getvariableattribute32()	832
16.69.2.47getvariableattribute64()	832
16.69.2.48getvariableattributei16()	832
16.69.2.49getvariableattributei32()	833
16.69.2.50getvariableattributei64()	833
16.69.2.51getvariableattributei8()	833
16.69.2.52getvariablefillvaluef32()	833
16.69.2.53getvariablefillvaluef64()	833
16.69.2.54getvariablefillvaluei16()	833
16.69.2.55getvariablefillvaluei32()	833
16.69.2.56getvariablefillvaluei64()	833
16.69.2.57getvariablefillvaluei8()	833
16.69.2.58hasattribute()	834
16.69.2.59nitncvariable()	834
16.69.2.60sunlimited()	834
16.69.2.61setattribute()	834
16.69.2.62setdata()	835
16.69.2.63setdata1df32()	835

16.69.2.64	setdata1df64()	835
16.69.2.65	setdata1di16()	835
16.69.2.66	setdata1di32()	835
16.69.2.67	setdata1di64()	835
16.69.2.68	setdata1di8()	836
16.69.2.69	setdata2df32()	836
16.69.2.70	setdata2df64()	836
16.69.2.71	setdata2di16()	836
16.69.2.72	setdata2di32()	836
16.69.2.73	setdata2di64()	836
16.69.2.74	setdata2di8()	836
16.69.2.75	setdata3df32()	836
16.69.2.76	setdata3df64()	836
16.69.2.77	setdata3di16()	837
16.69.2.78	setdata3di32()	837
16.69.2.79	setdata3di64()	837
16.69.2.80	setdata3di8()	837
16.69.2.81	setdata4df32()	837
16.69.2.82	setdata4df64()	837
16.69.2.83	setdata4di16()	837
16.69.2.84	setdata4di32()	837
16.69.2.85	setdata4di64()	837
16.69.2.86	setdata4di8()	838
16.69.2.87	setdata5df32()	838
16.69.2.88	setdata5df64()	838
16.69.2.89	setdata5di16()	838
16.69.2.90	setdata5di32()	838
16.69.2.91	setdata5di64()	838
16.69.2.92	setdata5di8()	838
16.69.2.93	setdatascalarf32()	838
16.69.2.94	setdatascalarf64()	838
16.69.2.95	setdatascalari16()	839
16.69.2.96	setdatascalari32()	839
16.69.2.97	setdatascalari64()	839
16.69.2.98	setdatascalarl8()	839
16.69.2.99	setfillvalue()	839
16.69.2.100	setvariableattributechar()	839
16.69.2.101	setvariableattributef32()	840
16.69.2.102	setvariableattributef64()	840
16.69.2.103	setvariableattributei16()	840

16.69.2.10 <code>Setvariableattributei32()</code>	840
16.69.2.10 <code>Setvariableattributei64()</code>	840
16.69.2.10 <code>Setvariableattributei8()</code>	840
16.69.2.10 <code>Setvariablefillvaluei32()</code>	840
16.69.2.10 <code>Setvariablefillvaluef64()</code>	840
16.69.2.10 <code>Setvariablefillvaluei16()</code>	840
16.69.2.10 <code>Setvariablefillvaluei32()</code>	841
16.69.2.11 <code>Setvariablefillvaluei64()</code>	841
16.69.2.11 <code>Setvariablefillvaluei8()</code>	841
16.69.3 Member Data Documentation	841
16.69.3.1 <code>dimension [1/2]</code>	841
16.69.3.2 <code>dimension [2/2]</code>	841
16.69.3.3 <code>id</code>	841
16.69.3.4 <code>netcdf</code>	841
16.69.3.5 <code>parent</code>	841
16.69.3.6 <code>s</code>	841
16.69.3.7 <code>the [1/2]</code>	842
16.69.3.8 <code>the [2/2]</code>	842
16.70 <code>mo_ncdf::ncvariable</code> Interface Reference	842
16.71 <code>mo_utils::ne</code> Interface Reference	842
16.71.1 Member Function/Subroutine Documentation	842
16.71.1.1 <code>notequal_dp()</code>	842
16.71.1.2 <code>notequal_sp()</code>	842
16.72 <code>mo_orderpack::nearless</code> Interface Reference	843
16.72.1 Member Function/Subroutine Documentation	843
16.72.1.1 <code>d_nearless()</code>	843
16.72.1.2 <code>i_nearless()</code>	843
16.72.1.3 <code>r_nearless()</code>	843
16.73 <code>mo_spatialsimilarity::nndv</code> Interface Reference	843
16.73.1 Detailed Description	843
16.73.2 Member Function/Subroutine Documentation	845
16.73.2.1 <code>nndv_dp()</code>	845
16.73.2.2 <code>nndv_sp()</code>	845
16.74 <code>mo_utils::notequal</code> Interface Reference	845
16.74.1 Member Function/Subroutine Documentation	845
16.74.1.1 <code>notequal_dp()</code>	845
16.74.1.2 <code>notequal_sp()</code>	846
16.75 <code>mo_errormeasures::nse</code> Interface Reference	846
16.75.1 Member Function/Subroutine Documentation	846
16.75.1.1 <code>nse_dp_1d()</code>	846

16.75.1.2 <code>nse_dp_2d()</code>	846
16.75.1.3 <code>nse_dp_3d()</code>	846
16.75.1.4 <code>nse_sp_1d()</code>	847
16.75.1.5 <code>nse_sp_2d()</code>	847
16.75.1.6 <code>nse_sp_3d()</code>	847
16.76 <code>mo_string_utils::num2str</code> Interface Reference	847
16.76.1 Detailed Description	847
16.76.2 Member Function/Subroutine Documentation	848
16.76.2.1 <code>dp2str()</code>	848
16.76.2.2 <code>i42str()</code>	848
16.76.2.3 <code>i82str()</code>	848
16.76.2.4 <code>log2str()</code>	848
16.76.2.5 <code>sp2str()</code>	849
16.77 <code>mo_string_utils::numarray2str</code> Interface Reference	849
16.77.1 Detailed Description	849
16.77.2 Member Function/Subroutine Documentation	849
16.77.2.1 <code>i4array2str()</code>	849
16.78 <code>mo_optimization_utils::objective_interface</code> Interface Reference	850
16.78.1 Constructor & Destructor Documentation	850
16.78.1.1 <code>objective_interface()</code>	850
16.79 <code>mo_orderpack::omedian</code> Interface Reference	850
16.79.1 Member Function/Subroutine Documentation	850
16.79.1.1 <code>d_median()</code>	850
16.79.1.2 <code>i_median()</code>	850
16.79.1.3 <code>r_median()</code>	851
16.80 <code>mo_mrm_write_fluxes_states::outputdataset</code> Interface Reference	851
16.80.1 Member Function/Subroutine Documentation	852
16.80.1.1 <code>close()</code>	852
16.80.1.2 <code>updatedataset()</code>	852
16.80.1.3 <code>writetimestep()</code>	852
16.80.2 Member Data Documentation	852
16.80.2.1 <code>all</code>	852
16.80.2.2 <code>basin</code>	852
16.80.2.3 <code>count</code>	853
16.80.2.4 <code>counter</code>	853
16.80.2.5 <code>created</code>	853
16.80.2.6 <code>ibasin</code>	853
16.80.2.7 <code>id</code>	853
16.80.2.8 <code>nc</code>	853
16.80.2.9 <code>ncdataset</code>	853

16.80.2.10steps	853
16.80.2.11store	853
16.80.2.12time	854
16.80.2.13to	854
16.80.2.14variables	854
16.80.2.15vars	854
16.80.2.16write	854
16.80.2.17written	854
16.81mo_write_fluxes_states::outputdataset Interface Reference	855
16.81.1 Member Function/Subroutine Documentation	856
16.81.1.1 close()	856
16.81.1.2 updatedataset()	856
16.81.1.3 writetimestep()	856
16.81.2 Member Data Documentation	856
16.81.2.1 all	856
16.81.2.2 basin	856
16.81.2.3 count	857
16.81.2.4 counter	857
16.81.2.5 created	857
16.81.2.6 ibasin	857
16.81.2.7 id	857
16.81.2.8 nc	857
16.81.2.9 ncdataset	857
16.81.2.10steps	857
16.81.2.11store	857
16.81.2.12time	858
16.81.2.13to	858
16.81.2.14variables	858
16.81.2.15vars	858
16.81.2.16write	858
16.81.2.17written	858
16.82mo_write_fluxes_states::outputvariable Interface Reference	859
16.82.1 Member Function/Subroutine Documentation	860
16.82.1.1 updatevariable()	860
16.82.1.2 writevariabletimestep()	860
16.82.2 Member Data Documentation	860
16.82.2.1 average	860
16.82.2.2 avg	860
16.82.2.3 before	861
16.82.2.4 between	861

16.82.2.5 calls	861
16.82.2.6 contains	861
16.82.2.7 count	861
16.82.2.8 counter	861
16.82.2.9 data [1/2]	861
16.82.2.10data [2/2]	861
16.82.2.11mask	861
16.82.2.12hc	862
16.82.2.13ncdataset	862
16.82.2.14number	862
16.82.2.15of	862
16.82.2.16reconstruct	862
16.82.2.17store	862
16.82.2.18he [1/3]	862
16.82.2.19he [2/3]	862
16.82.2.20he [3/3]	862
16.82.2.21to	863
16.82.2.22updatevariable	863
16.82.2.23variable	863
16.82.2.24which	863
16.82.2.25writes	863
16.82.2.26writing	863
16.83mo_mrm_write_fluxes_states::outputvariable Interface Reference	864
16.83.1 Member Function/Subroutine Documentation	865
16.83.1.1 updatevariable()	865
16.83.1.2 writevariabletimestep()	865
16.83.2 Member Data Documentation	865
16.83.2.1 average	865
16.83.2.2 avg	865
16.83.2.3 before	866
16.83.2.4 between	866
16.83.2.5 calls	866
16.83.2.6 contains	866
16.83.2.7 count	866
16.83.2.8 counter	866
16.83.2.9 data [1/2]	866
16.83.2.10data [2/2]	866
16.83.2.11mask	866
16.83.2.12hc	867
16.83.2.13ncdataset	867

16.83.2.14	number	867
16.83.2.15	of	867
16.83.2.16	reconstruct	867
16.83.2.17	store	867
16.83.2.18	he [1/3]	867
16.83.2.19	he [2/3]	867
16.83.2.20	he [3/3]	867
16.83.2.21	to	868
16.83.2.22	updatevariable	868
16.83.2.23	variable	868
16.83.2.24	which	868
16.83.2.25	writes	868
16.83.2.26	writing	868
16.84	mo_append::paste Interface Reference	868
16.84.1	Detailed Description	869
16.84.2	Member Function/Subroutine Documentation	869
16.84.2.1	paste_char_m_m()	869
16.84.2.2	paste_char_m_s()	869
16.84.2.3	paste_char_m_v()	870
16.84.2.4	paste_dp_m_m()	870
16.84.2.5	paste_dp_m_s()	870
16.84.2.6	paste_dp_m_v()	870
16.84.2.7	paste_i4_m_m()	870
16.84.2.8	paste_i4_m_s()	870
16.84.2.9	paste_i4_m_v()	871
16.84.2.10	paste_i8_m_m()	871
16.84.2.11	paste_i8_m_s()	871
16.84.2.12	paste_i8_m_v()	871
16.84.2.13	paste_lgt_m_m()	871
16.84.2.14	paste_lgt_m_s()	871
16.84.2.15	paste_lgt_m_v()	871
16.84.2.16	paste_sp_m_m()	872
16.84.2.17	paste_sp_m_s()	872
16.84.2.18	paste_sp_m_v()	872
16.85	mo_spatialsimilarity::pd Interface Reference	872
16.85.1	Detailed Description	872
16.85.2	Member Function/Subroutine Documentation	873
16.85.2.1	pd_dp()	874
16.85.2.2	pd_sp()	874
16.86	mo_percentile::percentile Interface Reference	874

16.86.1 Member Function/Subroutine Documentation	874
16.86.1.1 percentile_0d_dp()	874
16.86.1.2 percentile_0d_sp()	874
16.86.1.3 percentile_1d_dp()	875
16.86.1.4 percentile_1d_sp()	875
16.87 mo_common_variables::period Type Reference	875
16.87.1 Member Data Documentation	876
16.87.1.1 dend	876
16.87.1.2 dstart	876
16.87.1.3 julend	876
16.87.1.4 julstart	876
16.87.1.5 mend	876
16.87.1.6 mstart	876
16.87.1.7 nobs	876
16.87.1.8 yend	877
16.87.1.9 ystart	877
16.88 mo_percentile::qmedian Interface Reference	877
16.88.1 Member Function/Subroutine Documentation	877
16.88.1.1 qmedian_dp()	877
16.88.1.2 qmedian_sp()	877
16.89 mo_orderpack::rapknr Interface Reference	877
16.89.1 Member Function/Subroutine Documentation	877
16.89.1.1 d_rapknr()	878
16.89.1.2 i_rapknr()	878
16.89.1.3 r_rapknr()	878
16.90 mo_read_spatial_data::read_spatial_data_ascii Interface Reference	878
16.90.1 Detailed Description	878
16.90.2 Member Function/Subroutine Documentation	879
16.90.2.1 read_spatial_data_ascii_dp()	879
16.90.2.2 read_spatial_data_ascii_i4()	879
16.91 mo_corr::realft Interface Reference	880
16.91.1 Member Function/Subroutine Documentation	880
16.91.1.1 realft_dp()	880
16.91.1.2 realft_sp()	881
16.92 mo_orderpack::refpar Interface Reference	881
16.92.1 Member Function/Subroutine Documentation	881
16.92.1.1 d_refpar()	881
16.92.1.2 i_refpar()	881
16.92.1.3 r_refpar()	881
16.93 mo_orderpack::refsor Interface Reference	882

16.93.1 Member Function/Subroutine Documentation	882
16.93.1.1 d_refsor()	882
16.93.1.2 i_refsor()	882
16.93.1.3 r_refsor()	882
16.94 mo_orderpack::rinpar Interface Reference	882
16.94.1 Member Function/Subroutine Documentation	882
16.94.1.1 d_rinpar()	882
16.94.1.2 i_rinpar()	883
16.94.1.3 r_rinpar()	883
16.95 mo_errormeasures::rmse Interface Reference	883
16.95.1 Member Function/Subroutine Documentation	883
16.95.1.1 rmse_dp_1d()	883
16.95.1.2 rmse_dp_2d()	883
16.95.1.3 rmse_dp_3d()	884
16.95.1.4 rmse_sp_1d()	884
16.95.1.5 rmse_sp_2d()	884
16.95.1.6 rmse_sp_3d()	884
16.96 mo_orderpack::rnkpar Interface Reference	884
16.96.1 Member Function/Subroutine Documentation	884
16.96.1.1 d_rnkpar()	885
16.96.1.2 i_rnkpar()	885
16.96.1.3 r_rnkpar()	885
16.97 mo_errormeasures::sae Interface Reference	885
16.97.1 Member Function/Subroutine Documentation	885
16.97.1.1 sae_dp_1d()	885
16.97.1.2 sae_dp_2d()	886
16.97.1.3 sae_dp_3d()	886
16.97.1.4 sae_sp_1d()	886
16.97.1.5 sae_sp_2d()	886
16.97.1.6 sae_sp_3d()	886
16.98 mo_julian::setcalendar Interface Reference	886
16.98.1 Member Function/Subroutine Documentation	887
16.98.1.1 setcalendarinteger()	887
16.98.1.2 setcalendarstring()	887
16.99 mo_moment::skewness Interface Reference	887
16.99.1 Member Function/Subroutine Documentation	888
16.99.1.1 skewness_dp()	888
16.99.1.2 skewness_sp()	888
16.100 mo_mpr_global_variables::soiltype Type Reference	888
16.100.1 Member Data Documentation	889

16.100.1.1	clay	889
16.100.1.2	db	889
16.100.1.3	dbm	889
16.100.1.4	depth	889
16.100.1.5	d	889
16.100.1.6	s_present	890
16.100.1.7	ks	890
16.100.1.8	d	890
16.100.1.9	nhorizons	890
16.100.1.10	tillhorizons	890
16.100.1.11	zdepth	890
16.100.1.12	and	890
16.100.1.13	betafc	890
16.100.1.14	metafc_till	890
16.100.1.15	etapw	891
16.100.1.16	etapw_till	891
16.100.1.17	etas	891
16.100.1.18	etas_till	891
16.100.1.19	d	891
16.100.1.20	d	891
16.10	no_orderpack::sort Interface Reference	891
16.101.	Detailed Description	891
16.101.1	Member Function/Subroutine Documentation	894
16.101.2.1	d_refsor()	894
16.101.2.2	_refsor()	894
16.101.2.3	_refsor()	894
16.102	no_orderpack::sort_index Interface Reference	894
16.102.1	Member Function/Subroutine Documentation	894
16.102.1.1	sort_index_dp()	895
16.102.1.2	sort_index_i4()	895
16.102.1.3	sort_index_sp()	895
16.103	no_spatial_agg_disagg_forcing::spatial_aggregation Interface Reference	895
16.103.1	Detailed Description	895
16.103.1.1	Member Function/Subroutine Documentation	895
16.103.2.1	spatial_aggregation_3d()	896
16.103.2.2	spatial_aggregation_4d()	896
16.104	no_spatial_agg_disagg_forcing::spatial_disaggregation Interface Reference	896
16.104.1	Detailed Description	896
16.104.1.1	Member Function/Subroutine Documentation	897
16.104.2.1	spatial_disaggregation_3d()	897

16.104.2.2spatial_disaggregation_4d()	897
16.105no_utils::special_value Interface Reference	897
16.105.1Detailed Description	897
16.105.2Member Function/Subroutine Documentation	898
16.105.2.1special_value_dp()	898
16.105.2.2special_value_sp()	899
16.106no_kind::sprs2_dp Type Reference	899
16.106.1Detailed Description	899
16.106.2Member Data Documentation	899
16.106.2.1irow	899
16.106.2.2col	900
16.106.2.3en	900
16.106.2.4n	900
16.106.2.5val	900
16.107no_kind::sprs2_sp Type Reference	900
16.107.1Detailed Description	901
16.107.2Member Data Documentation	901
16.107.2.1irow	901
16.107.2.2col	901
16.107.2.3en	901
16.107.2.4n	901
16.107.2.5val	901
16.108no_errormeasures::sse Interface Reference	901
16.108.1Member Function/Subroutine Documentation	902
16.108.1.1sse_dp_1d()	902
16.108.1.2sse_dp_2d()	902
16.108.1.3sse_dp_3d()	902
16.108.1.4sse_sp_1d()	902
16.108.1.5sse_sp_2d()	902
16.108.1.6sse_sp_3d()	902
16.109no_standard_score::standard_score Interface Reference	903
16.109.1Detailed Description	903
16.109.2Member Function/Subroutine Documentation	904
16.109.2.1standard_score_dp()	904
16.109.2.2standard_score_sp()	904
16.110no_moment::stddev Interface Reference	904
16.110.1Member Function/Subroutine Documentation	904
16.110.1.1stddev_dp()	904
16.110.1.2stddev_sp()	904
16.111no_corr::swap Interface Reference	905

16.111.1Member Function/Subroutine Documentation	905
16.111.1.1swap_1d_dpc()	905
16.111.1.2swap_1d_spc()	905
16.112no_utils::swap Interface Reference	905
16.112.1Detailed Description	905
16.112.2Member Function/Subroutine Documentation	906
16.112.2.1swap_vec_dp()	906
16.112.2.2swap_vec_i4()	906
16.112.2.3swap_vec_sp()	906
16.112.2.4swap_xy_dp()	906
16.112.2.5swap_xy_i4()	907
16.112.2.6swap_xy_sp()	907
16.113no_global_variables::twsstructure Type Reference	907
16.113.1Member Data Documentation	907
16.113.1.1basinid	907
16.113.1.2name	908
16.113.1.3ws	908
16.114no_orderpack::uniinv Interface Reference	908
16.114.1Member Function/Subroutine Documentation	908
16.114.1.1d_uniinv()	908
16.114.1.2_uniinv()	908
16.114.1.3_uniinv()	908
16.115no_orderpack::unipar Interface Reference	909
16.115.1Member Function/Subroutine Documentation	909
16.115.1.1d_unipar()	909
16.115.1.2_unipar()	909
16.115.1.3_unipar()	909
16.116no_orderpack::unirnk Interface Reference	909
16.116.1Member Function/Subroutine Documentation	909
16.116.1.1d_unirnk()	910
16.116.1.2_unirnk()	910
16.116.1.3_unirnk()	910
16.117no_orderpack::unista Interface Reference	910
16.117.1Member Function/Subroutine Documentation	910
16.117.1.1d_unista()	910
16.117.1.2_unista()	910
16.117.1.3_unista()	911
16.118no_mpr_restart::unpack_field_and_write Interface Reference	911
16.118.1Detailed Description	911
16.118.2Member Function/Subroutine Documentation	911

16.118.2.1unpack_field_and_write_1d_dp()	911
16.118.2.2unpack_field_and_write_1d_i4()	912
16.118.2.3unpack_field_and_write_2d_dp()	912
16.118.2.4unpack_field_and_write_3d_dp()	912
16.119no_restart::unpack_field_and_write Interface Reference	912
16.119.1Detailed Description	913
16.119.2Member Function/Subroutine Documentation	913
16.119.2.1unpack_field_and_write_1d_dp()	913
16.119.2.2unpack_field_and_write_1d_i4()	913
16.119.2.3unpack_field_and_write_2d_dp()	914
16.119.2.4unpack_field_and_write_3d_dp()	914
16.120no_orderpack::valmed Interface Reference	914
16.120.1Member Function/Subroutine Documentation	914
16.120.1.1d_valmed()	914
16.120.1.2_valmed()	915
16.120.1.3_valmed()	915
16.121no_orderpack::valnth Interface Reference	915
16.121.1Member Function/Subroutine Documentation	915
16.121.1.1d_valnth()	915
16.121.1.2_valnth()	915
16.121.1.3_valnth()	915
16.122no_ncwrite::var2nc Interface Reference	916
16.122.1Detailed Description	916
16.122.2Member Function/Subroutine Documentation	917
16.122.2.1var2nc_1d_dp()	917
16.122.2.2var2nc_1d_i4()	918
16.122.2.3var2nc_1d_sp()	918
16.122.2.4var2nc_2d_dp()	918
16.122.2.5var2nc_2d_i4()	919
16.122.2.6var2nc_2d_sp()	919
16.122.2.7var2nc_3d_dp()	919
16.122.2.8var2nc_3d_i4()	919
16.122.2.9var2nc_3d_sp()	920
16.122.2.10var2nc_4d_dp()	920
16.122.2.11var2nc_4d_i4()	920
16.122.2.12var2nc_4d_sp()	921
16.122.2.13var2nc_5d_dp()	921
16.122.2.14var2nc_5d_i4()	921
16.122.2.15var2nc_5d_sp()	922
16.123no_ncwrite::variable Type Reference	923

16.123.1Member Data Documentation	924
16.123.1.1att	924
16.123.1.2count	924
16.123.1.3dimids	924
16.123.1.4dimtypes	925
16.123.1.5g0_b	925
16.123.1.6g0_d	925
16.123.1.7g0_f	925
16.123.1.8g0_i	925
16.123.1.9g1_b	925
16.123.1.101_d	925
16.123.1.1g1_f	925
16.123.1.1g1_i	925
16.123.1.1g2_b	926
16.123.1.1g2_d	926
16.123.1.1g2_f	926
16.123.1.1g2_i	926
16.123.1.1g3_b	926
16.123.1.1g3_d	926
16.123.1.1g3_f	926
16.123.1.2g3_i	926
16.123.1.2g4_b	926
16.123.1.2g4_d	927
16.123.1.2g4_f	927
16.123.1.2g4_i	927
16.123.1.25ame	927
16.123.1.26att	927
16.123.1.27dimids	927
16.123.1.28vlvs	927
16.123.1.29subs	927
16.123.1.30start	927
16.123.1.31unlimited	928
16.123.1.32arid	928
16.123.1.33flag	928
16.123.1.34type	928
16.124no_moment::variance Interface Reference	928
16.124.Member Function/Subroutine Documentation	928
16.124.1.1variance_dp()	928
16.124.1.2variance_sp()	928
16.125no_errormeasures::wnse Interface Reference	929

16.125.1 Member Function/Subroutine Documentation	929
16.125.1.1wnse_dp_1d()	929
16.125.1.2wnse_dp_2d()	929
16.125.1.3wnse_dp_3d()	929
16.125.1.4wnse_sp_1d()	929
16.125.1.5wnse_sp_2d()	929
16.125.1.6wnse_sp_3d()	930
16.126mo_xor4096::xor4096 Interface Reference	930
16.126.1 Member Function/Subroutine Documentation	930
16.126.1.1xor4096d_0d()	930
16.126.1.2xor4096d_1d()	930
16.126.1.3xor4096f_0d()	931
16.126.1.4xor4096f_1d()	931
16.126.1.5xor4096l_0d()	931
16.126.1.6xor4096l_1d()	931
16.126.1.7xor4096s_0d()	931
16.126.1.8xor4096s_1d()	931
16.127mo_xor4096::xor4096g Interface Reference	932
16.127.1 Member Function/Subroutine Documentation	932
16.127.1.1xor4096gd_0d()	932
16.127.1.2xor4096gd_1d()	932
16.127.1.3xor4096gf_0d()	932
16.127.1.4xor4096gf_1d()	932
17 File Documentation	933
17.1 1-main.dox File Reference	933
17.2 2-get_started.dox File Reference	933
17.3 3-data_preparation.dox File Reference	933
17.4 4-visualise_out.dox File Reference	933
17.5 5-calibration.dox File Reference	933
17.6 6-style_guide.dox File Reference	933
17.7 7-test_basin.dox File Reference	933
17.8 8-protocols_for_setup_new_mHM_basin.dox File Reference	933
17.9 DEPENDENCIES.md File Reference	933
17.10mhm_driver.f90 File Reference	933
17.10.1 Function/Subroutine Documentation	933
17.10.1.1 mhm_driver()	934
17.11mhm_papers.md File Reference	936
17.12mo_anneal.f90 File Reference	936
17.13mo_append.f90 File Reference	937

17.14mo_canopy_interc.f90 File Reference	938
17.15mo_common_constants.f90 File Reference	938
17.16mo_common_file.f90 File Reference	939
17.17mo_common_functions.f90 File Reference	939
17.18mo_common_mHM_mRM_file.f90 File Reference	940
17.19mo_common_mHM_mRM_read_config.f90 File Reference	940
17.20mo_common_mHM_mRM_variables.f90 File Reference	940
17.21mo_common_read_config.f90 File Reference	941
17.22mo_common_read_data.f90 File Reference	941
17.23mo_common_restart.f90 File Reference	942
17.24mo_common_variables.f90 File Reference	942
17.25mo_constants.f90 File Reference	943
17.26mo_corr.f90 File Reference	945
17.27mo_dds.f90 File Reference	946
17.28mo_errormeasures.f90 File Reference	946
17.29mo_file.f90 File Reference	948
17.30mo_finish.f90 File Reference	949
17.31mo_global_variables.f90 File Reference	949
17.32mo_grid.f90 File Reference	951
17.33mo_init_states.f90 File Reference	951
17.34mo_julian.f90 File Reference	952
17.35mo_kind.f90 File Reference	953
17.36mo_linfit.f90 File Reference	954
17.37mo_mcmc.f90 File Reference	954
17.38mo_message.f90 File Reference	955
17.39mo_meteo_forcings.f90 File Reference	955
17.40mo_mhm.f90 File Reference	955
17.41mo_mhm_constants.f90 File Reference	956
17.42mo_mhm_eval.f90 File Reference	957
17.43mo_mhm_read_config.f90 File Reference	957
17.44mo_moment.f90 File Reference	957
17.45mo_mpr_constants.f90 File Reference	958
17.46mo_mpr_eval.f90 File Reference	959
17.47mo_mpr_file.f90 File Reference	960
17.48mo_mpr_global_variables.f90 File Reference	961
17.49mo_mpr_pet.f90 File Reference	962
17.50mo_mpr_read_config.f90 File Reference	963
17.51mo_mpr_restart.f90 File Reference	963
17.52mo_mpr_runoff.f90 File Reference	963
17.53mo_mpr_smhorizons.f90 File Reference	964

17.54mo_mpr_soilmoist.f90 File Reference	964
17.55mo_mpr_startup.f90 File Reference	964
17.56mo_mrm_constants.f90 File Reference	965
17.57mo_mrm_eval.f90 File Reference	965
17.58mo_mrm_file.f90 File Reference	965
17.59mo_mrm_global_variables.f90 File Reference	966
17.60mo_mrm_init.f90 File Reference	968
17.61mo_mrm_mpr.f90 File Reference	968
17.62mo_mrm_net_startup.f90 File Reference	969
17.63mo_mrm_objective_function_runoff.f90 File Reference	969
17.64mo_mrm_read_config.f90 File Reference	970
17.65mo_mrm_read_data.f90 File Reference	971
17.66mo_mrm_restart.f90 File Reference	971
17.67mo_mrm_routing.f90 File Reference	971
17.68mo_mrm_signatures.f90 File Reference	972
17.69mo_mrm_write.f90 File Reference	973
17.70mo_mrm_write_fluxes_states.f90 File Reference	973
17.71mo_multi_param_reg.f90 File Reference	974
17.72mo_ncread.f90 File Reference	975
17.73mo_ncwrite.f90 File Reference	976
17.74mo_netcdf.f90 File Reference	977
17.75mo_neutrons.f90 File Reference	980
17.76mo_nml.f90 File Reference	981
17.77mo_objective_function.f90 File Reference	981
17.78mo_optimization.f90 File Reference	982
17.79mo_optimization_utils.f90 File Reference	982
17.80mo_orderpack.f90 File Reference	983
17.81mo_percentile.f90 File Reference	985
17.82mo_pet.f90 File Reference	985
17.83mo_prepare_gridded_lai.f90 File Reference	986
17.84mo_read_forcing_nc.f90 File Reference	986
17.85mo_read_latlon.f90 File Reference	987
17.86mo_read_lut.f90 File Reference	987
17.87mo_read_optional_data.f90 File Reference	987
17.88mo_read_spatial_data.f90 File Reference	988
17.89mo_read_timeseries.f90 File Reference	988
17.90mo_read_wrapper.f90 File Reference	988
17.91mo_restart.f90 File Reference	989
17.92mo_runoff.f90 File Reference	989
17.93mo_sce.f90 File Reference	989

17.93.1 Function/Subroutine Documentation	990
17.93.1.1 set_optional()	990
17.93.1.2 write_best_final()	990
17.93.1.3 write_best_intermediate()	991
17.93.1.4 write_population()	991
17.93.1.5 write_termination_case()	992
17.94mo_set_ncdf_outputs.f90 File Reference	992
17.95mo_snow_accum_melt.f90 File Reference	992
17.96mo_soil_database.f90 File Reference	992
17.97mo_soil_moisture.f90 File Reference	993
17.98mo_spatial_agg_disagg_forcing.f90 File Reference	993
17.99mo_spatialsimilarity.f90 File Reference	994
17.10mo_standard_score.f90 File Reference	994
17.101mo_startup.f90 File Reference	994
17.102mo_string_utils.f90 File Reference	995
17.103mo_template.f90 File Reference	996
17.104mo_temporal_aggregation.f90 File Reference	996
17.105mo_temporal_disagg_forcing.f90 File Reference	997
17.106mo_timer.f90 File Reference	997
17.107mo_upscaling_operators.f90 File Reference	998
17.108mo_utils.f90 File Reference	998
17.109mo_write_ascii.f90 File Reference	999
17.110mo_write_fluxes_states.f90 File Reference	1000
17.111mo_xor4096.f90 File Reference	1001
17.112pr_driver.f90 File Reference	1001
17.112.1 Function/Subroutine Documentation	1001
17.112.1.1mpr_driver()	1002
17.112rm_driver.f90 File Reference	1003
17.113 Function/Subroutine Documentation	1004
17.113.1.1mrm_driver()	1004
17.114RELEASES.md File Reference	1005
Bibliography	1007
Index	1009

Chapter 1

Introduction to mHM

This chapter is divided in the following sections:

- [Short Description](#)
- [The Grid-based mHM Model](#)
- [Model Formulation](#)
- [The Multiscale Parameter Regionalization Technique](#)
- [The Parameter Estimation Problem](#)
- [Model Calibration](#)
- [Test basin](#)
- [Protocols](#)

1.1 Short Description

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

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

1.2 The Grid-based mHM Model

The mHM hydrologic model is based on numerical approximations of dominant hydrological processes that have been tested in various models: HBV [2], [10], [5] and VIC [11]. 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 [14] .

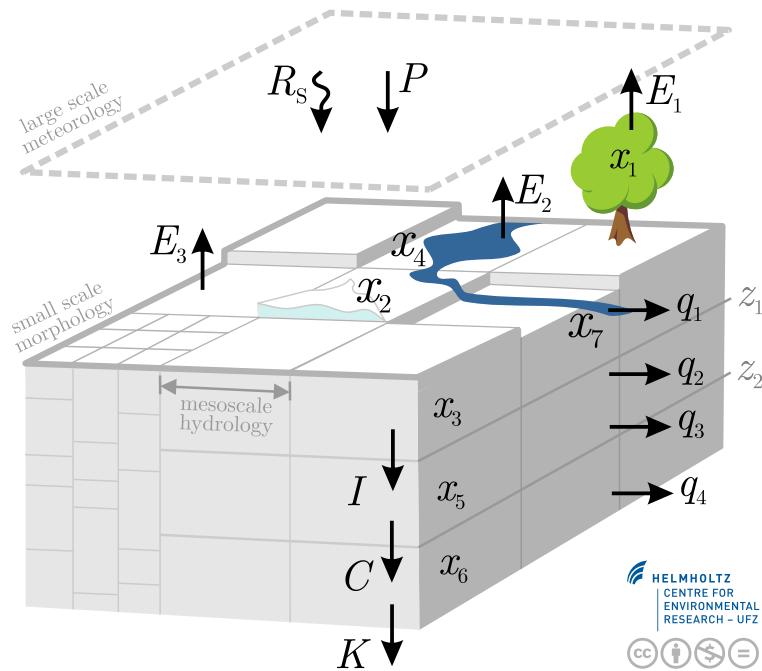


Figure 1.1: Typical mHM cell

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

- *Level-0*: Spatial discretization suitable to describe the main features of the terrain, the main soil characteristics (pedotop), and the land cover. The cell size at this level is denoted by ℓ_0 .
- *Level-1*: Spatial discretization used to describe dominant hydrological processes [4] at the mesoscale as well as the main geological formations of the basin. The cell size at this level is denoted by ℓ_1 .
- *Level-2*: Spatial discretization suitable to describe the variability of the meteorological forcings at the mesoscale, for example the formation of convective precipitation. The cell size at this level is denoted by ℓ_2 .

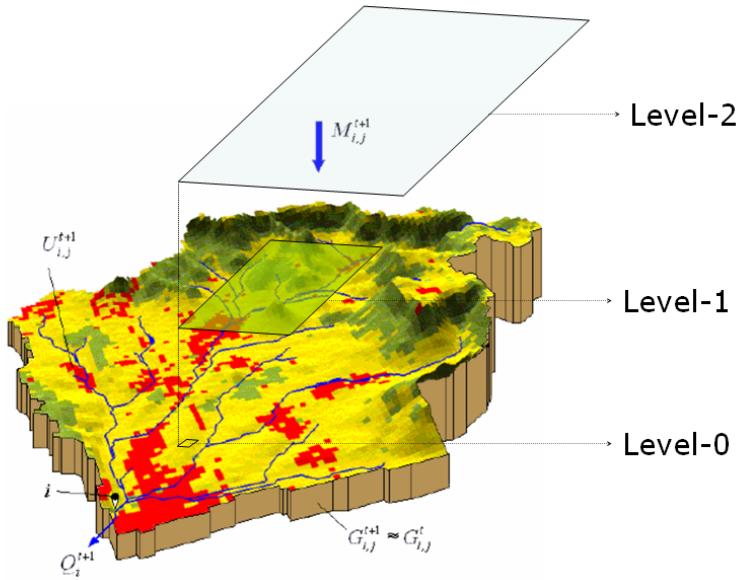


Figure 1.2: Hierarchy of data and modeling levels in mHM

1.3 Model Formulation

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

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

$$\forall i \in \Omega.$$

where

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

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

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

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

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, $l = 1, \dots, L$ (say $L = 3$), in the first layer, $0 \leq z \leq z_1$
t	Time index for each Δt interval
p^l	Overall influx fraction accounting for the impervious cover within a cell

1.4 The Multiscale Parameter Regionalization Technique

mHM requires at most 28 parameters (depending of the configuration) per cell to account for the spatial variability of the dominant hydrological processes at a mesoscale river basin. These *effective parameters* have to be estimated through calibration. Calibrating this model with a significant number of free parameters for every grid cell would lead to over-parameterization in a mesoscale catchment. This, in turn, would tend to increase the predictive uncertainty of the model due to the *equifinality* [3] of feasible solutions. Moreover, the high dimensionality of this optimization problem is also a daunting task for the state-of-the-art optimization algorithms [12]. To overcome this problem a

multiscale parameter regionalization (MPR) was employed in the mHM model [14].

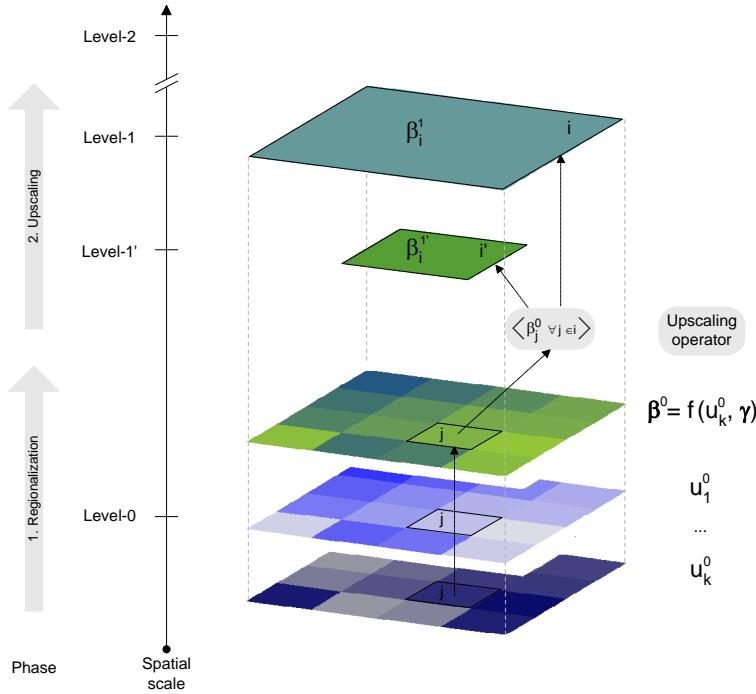


Figure 1.3: MPR

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:

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

where

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

Here $k = 1, \dots, K$ 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. $O_k(\bullet)_i$ denotes the kind of operator applied for the regionalization of the parameter k . Several types of operators were employed, e.g. majority \mathcal{M} , arithmetic mean \mathcal{A} , maximum difference \mathcal{D} , geometric mean \mathcal{G} , and harmonic mean \mathcal{H} . This table also shows the type of relationship employed for the transfer function and the predictors. By establishing such a relationships, the calibration algorithm finds good solutions for the transfer functions parameters ($s = 45$) instead of the model parameters for every grid cell. This, in turn, implies a great reduction of complexity since $K \times n \gg s$, where n denotes the total number of cells of a given basin at level-1.

1.5 The Parameter Estimation Problem

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

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

where

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

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

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

where $\|\cdot\|$ denotes a robust estimator. Many procedures to estimate global parameters are provided in mHM (e.g. simulated annealing [1], dynamically dimensioned search [15]). Other techniques can be found in the CHS Fortran Library.

1.6 Model Calibration

Good parameter sets for γ were identified with a split-sampling technique using an adaptive constrained optimization algorithm based on simulated annealing (SA) [1]. The overall model efficiency was estimated as a weighted combination of four estimators based on the Nash-Sutcliffe efficiency (NSE) between observed and calculated streamflows using three different time scales (daily, monthly and annual) as well as the logarithms of the streamflow to downplay the effects of the peak flows over the low flows [9]. These objective functions are denoted by ϕ_k , $k = 1, 4$. Every objective function should be normalized in the interval $[0, 1]$, with 1 representing the best possible 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 [8] and w_i denote the degree of importance of each objective. High values of p , say $p = 6$, should be chosen to avoid substitution of objective function values at low levels. In general, the estimators related to daily streamflows were twice as important as the long-term ones, thus $\{w_i\} = \{\frac{2}{4}, \frac{1}{4}, \frac{1}{4}, \frac{2}{4}\}$. The NSE for a given time interval t' is given by

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

where $\bar{y}_k(t')$ 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. Further 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 GitHub.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_ncdf.html

If not yet available you need to **install curl** on your 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):

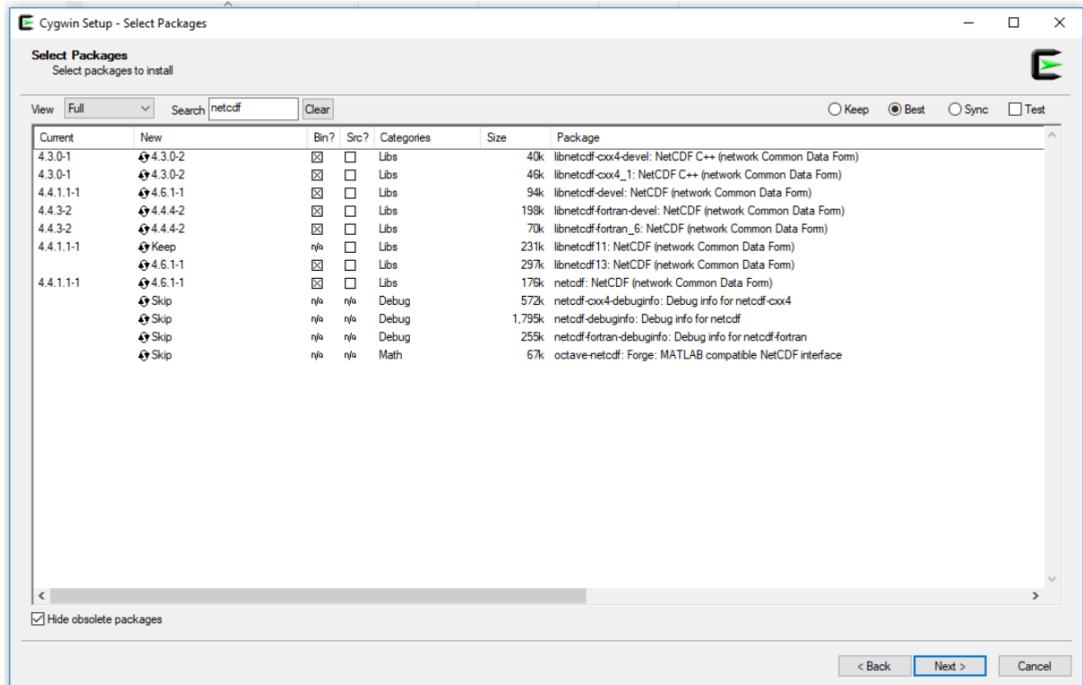


Figure 2.1: netcdf packages for CYGWIN

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

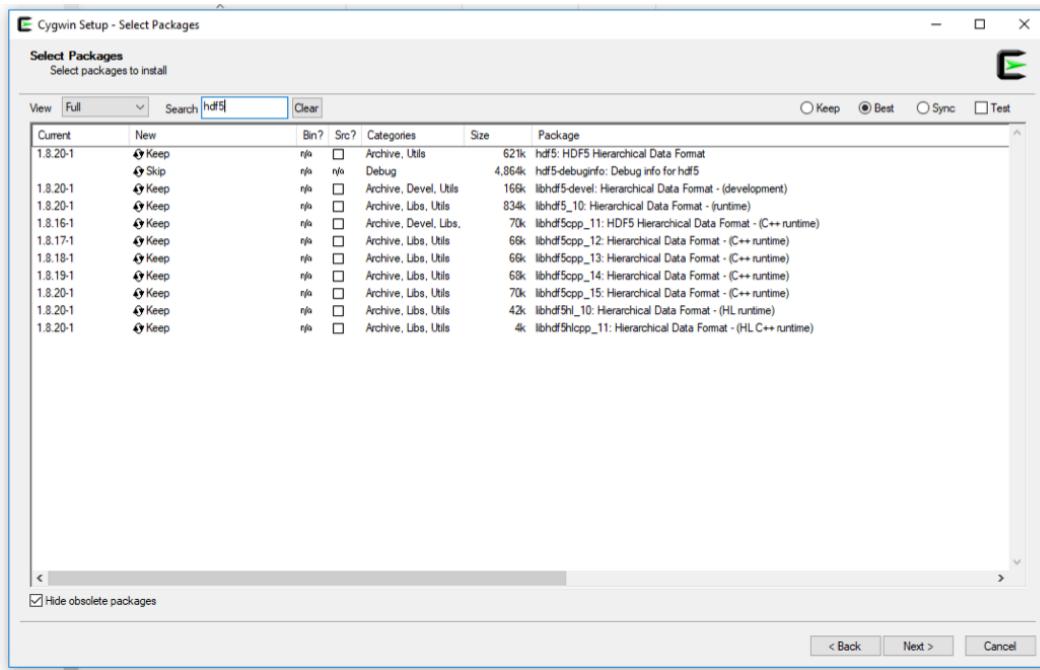


Figure 2.2: hdf5 packages for CYGWIN

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

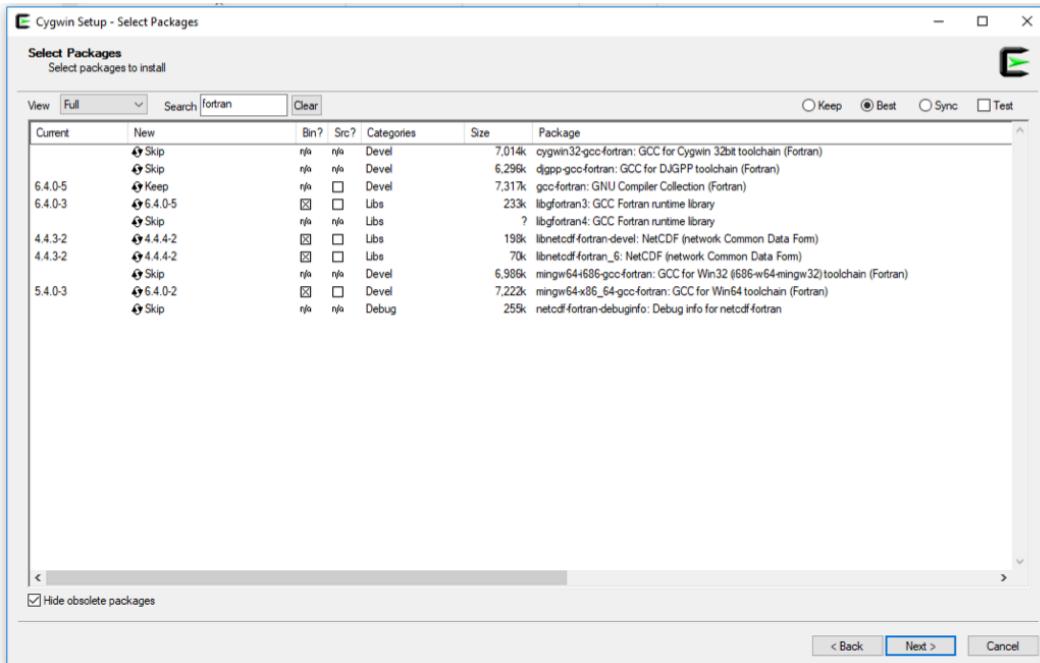


Figure 2.3: gfortran packages for CYGWIN

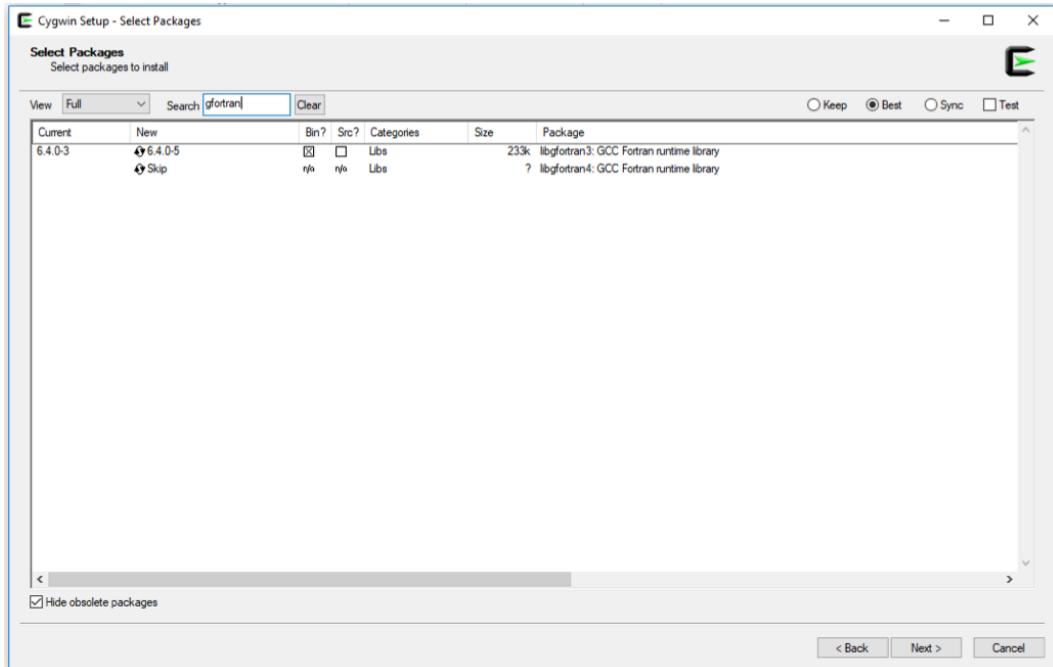


Figure 2.4: libgfortran packages for CYGWIN

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

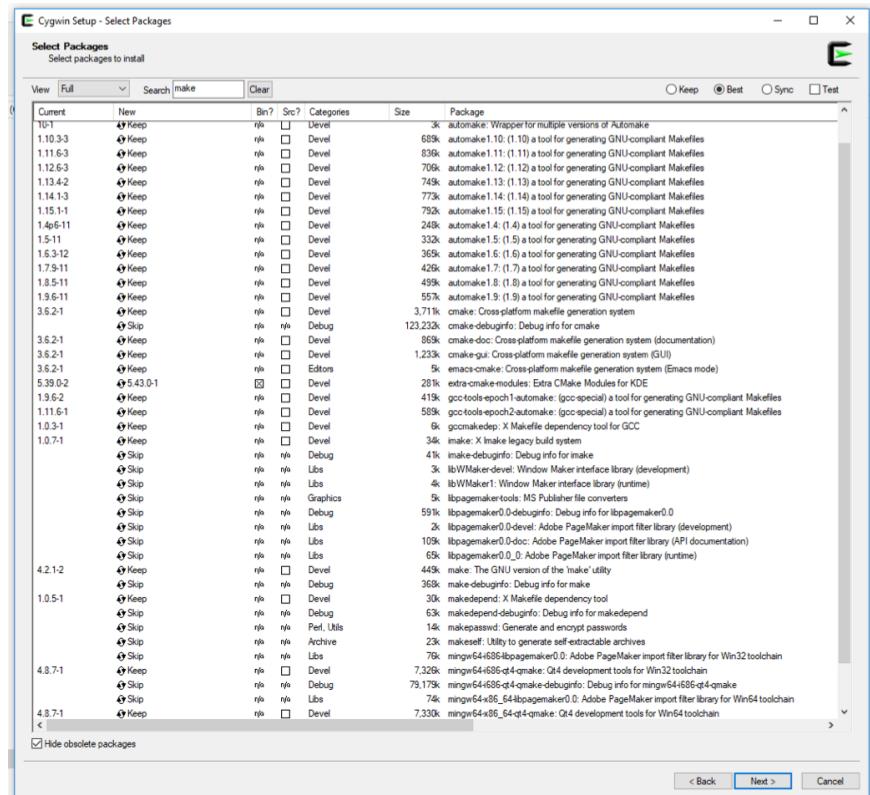


Figure 2.5: make packages for CYGWIN

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 `cygwin` folder in your environmental path ([fig_cygwin_path](#)).

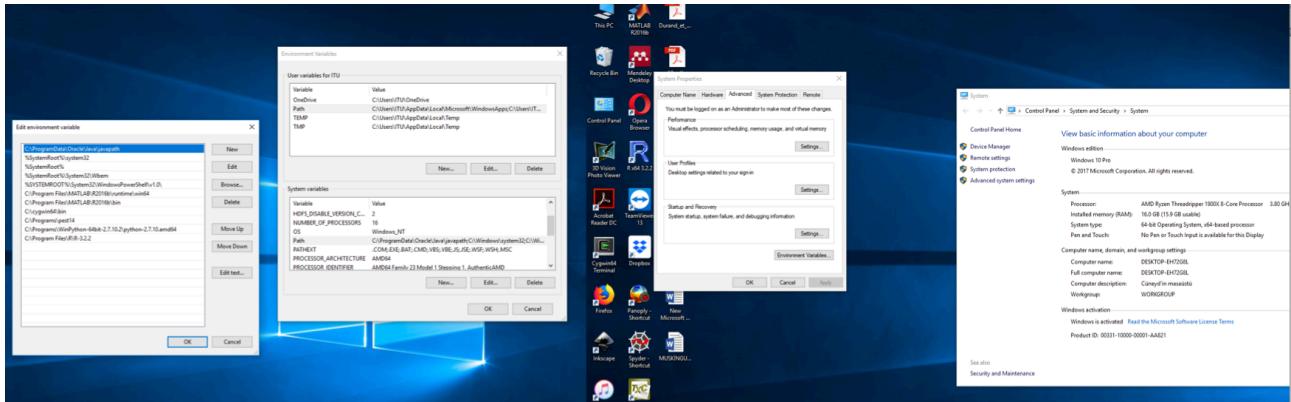


Figure 2.6: setup environmental path CYGWIN

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 your 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 be 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. Detailed 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 simultaneously 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_paraset.n.c` .
- **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.
- **InNSE:** logarithmic Nash-Sutcliffe Efficiency, recommended for fitting low flows.
- **0.5*(NSE+InNSE):** weights both NSE and InNSE 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 (InNSE), 3 (50% NSE+InNSE), 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 resources. 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, `path/to/FinalParams.out` and `path/to/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 resolution 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 following 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 needed:

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

- processCase(5) = 1 - Hargreaves-Samani equation
- processCase(5) = 2 - Priestley-Taylor equation
- processCase(5) = 3 - Penman-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 - Absolute vapor 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/COR0-landcover>), the Global Land Cover Map 2000 a dataset 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 measurements 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	m^3s^{-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 Extract 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 setrefdate,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 coronaer 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 WG↔S84 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 spatial resolutions connected to mHM. The three resolutions are the resolution 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_DIRECTORY]/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 following 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 environmental settings on every processing step outlined in the following paragraphs.

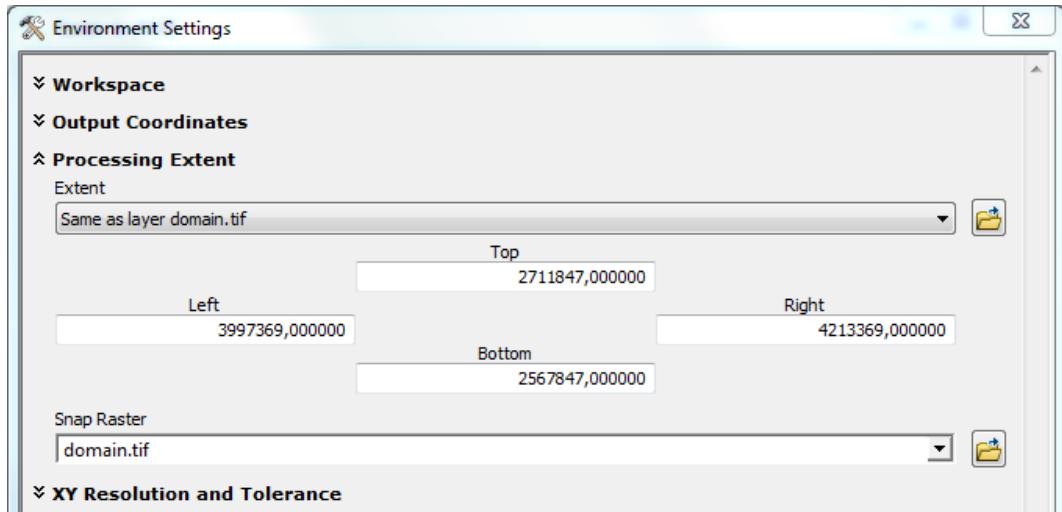


Figure 3.1: Button 'Environments...' in all Toolbox windows

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.

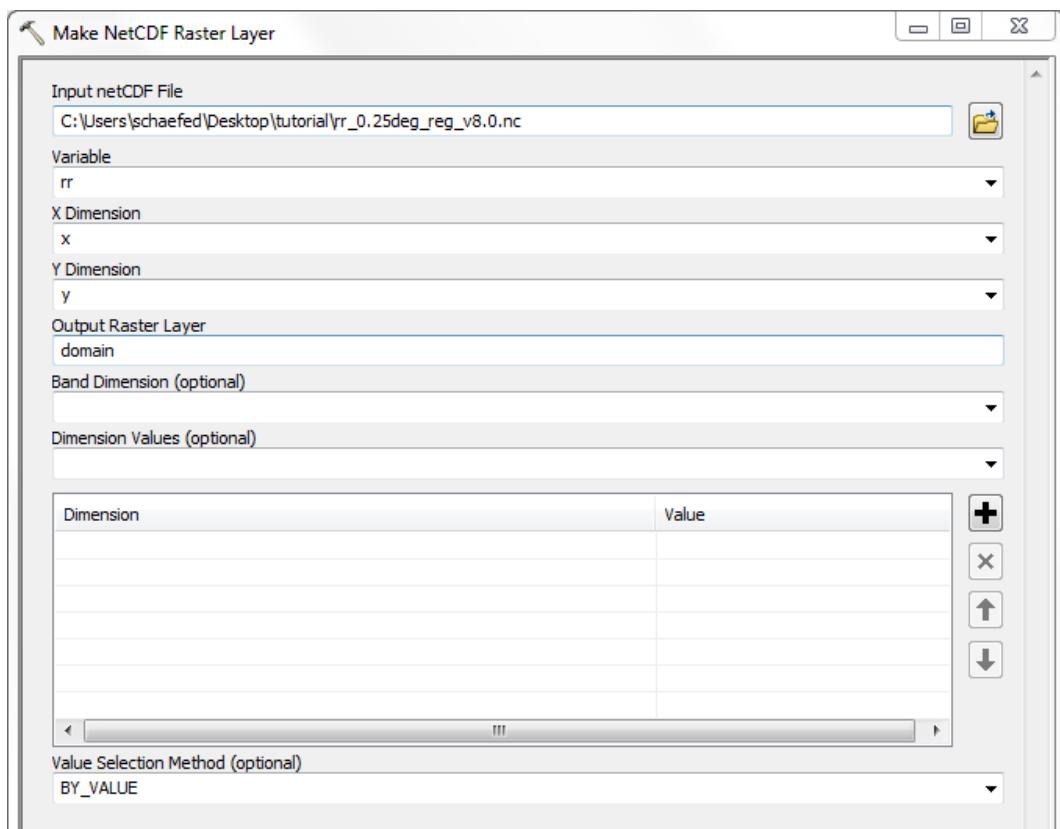


Figure 3.2: System Toolboxes -> Multidimension Tools -> Make NetCDF Raster Layer

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.

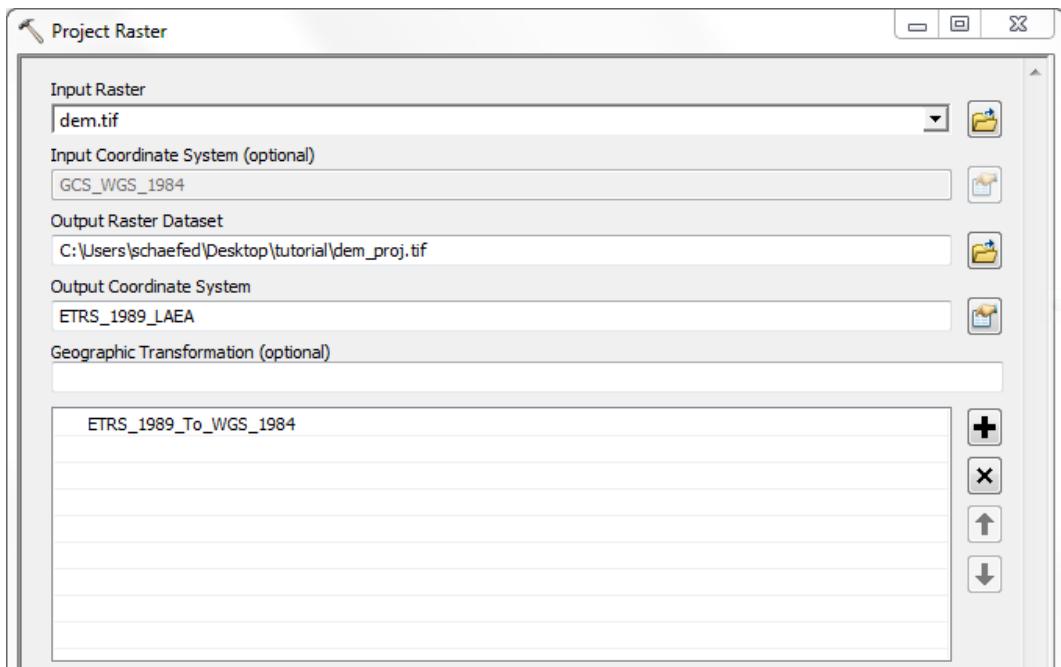


Figure 3.3: System Toolboxes -> Data Management Tools -> Projections and Transformations -> Raster -> Project Raster

- 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 appropriate 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.

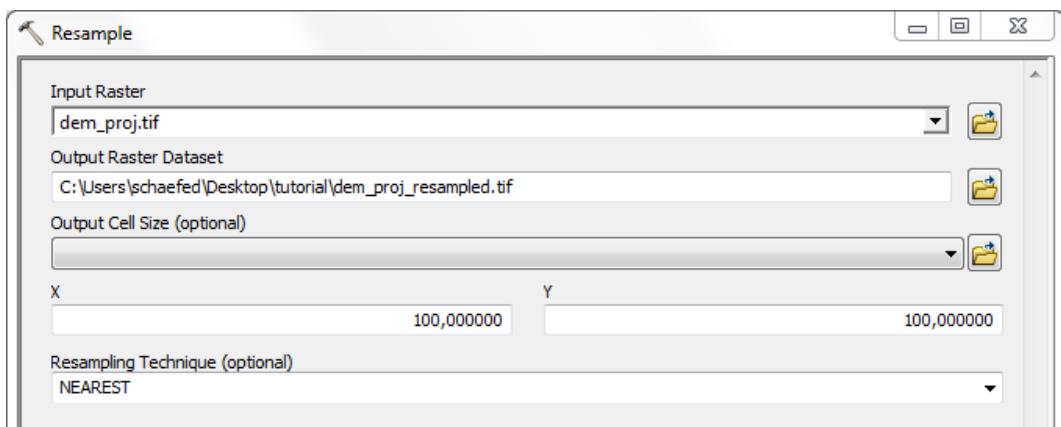


Figure 3.4: System Toolboxes -> Data Management Tools -> Raster -> Raster Processing -> Resample

- 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 defined 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' following 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

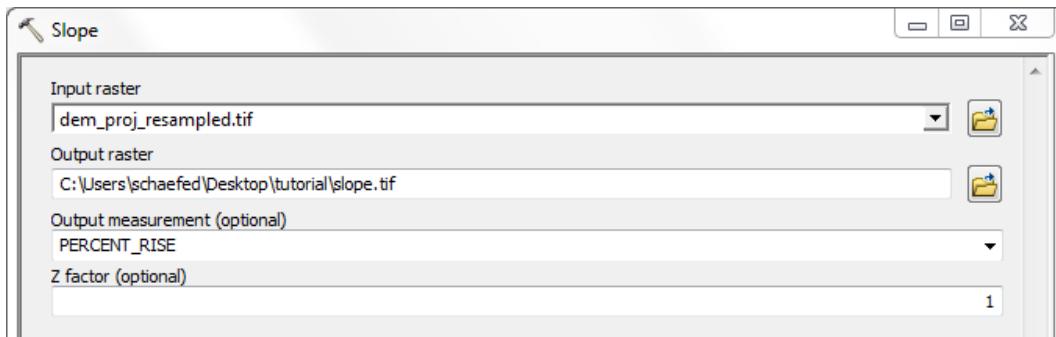


Figure 3.6: System Toolboxes -> Spatial Analyst Tools -> Surface -> Slope

3.5.3 Aspect map



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

3.5.4 Fill DEM sinks

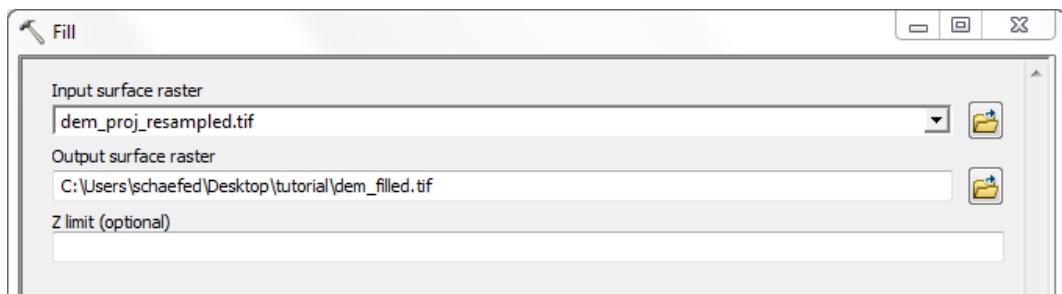


Figure 3.8: System Toolboxes -> Spatial Analyst Tools -> Hydrology -> Fill

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 Analyst 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 Direction

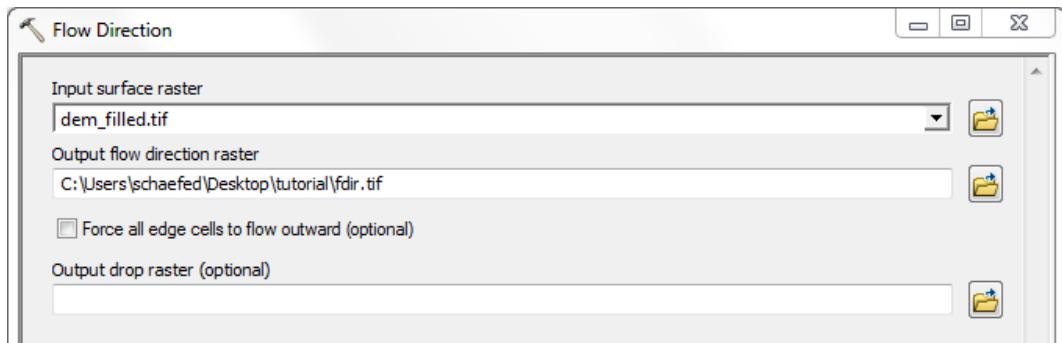


Figure 3.9: System Toolboxes -> Spatial Analyst Tools -> Hydrology -> Flow Direction

- Flow Accumulation

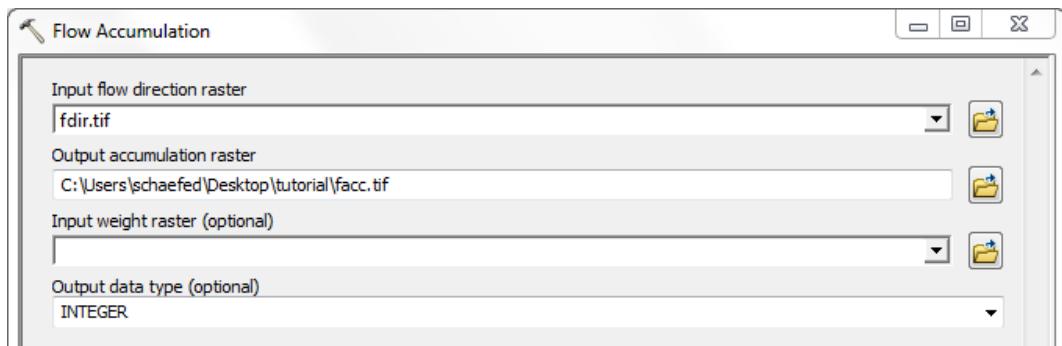


Figure 3.10: System Toolboxes -> Spatial Analyst Tools -> Hydrology -> Flow Accumulation

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>.

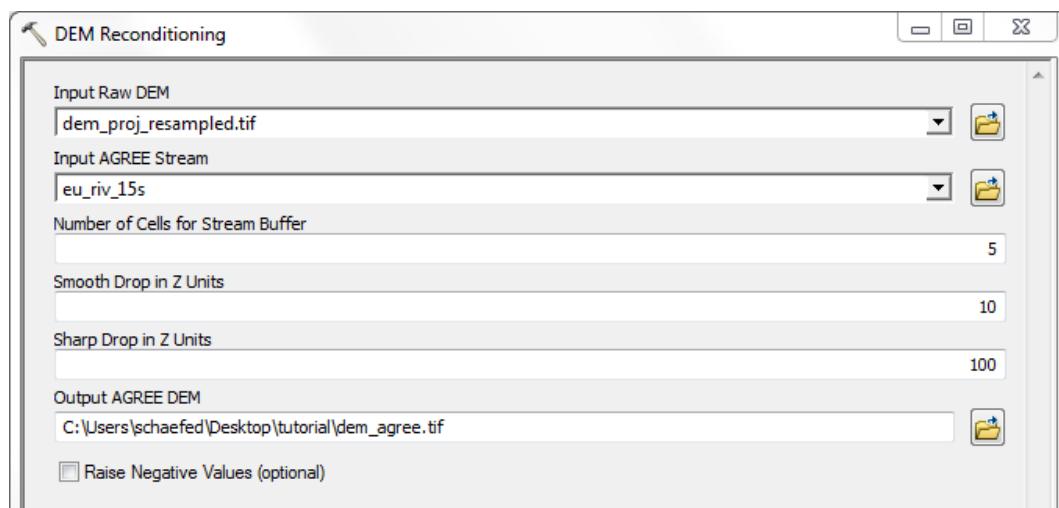


Figure 3.11: System Toolboxes -> Arc Hydro Tools -> Terrain Preprocessing -> DEM Reconditioning

- Fill the sinks in the resulting reconditioned DEM

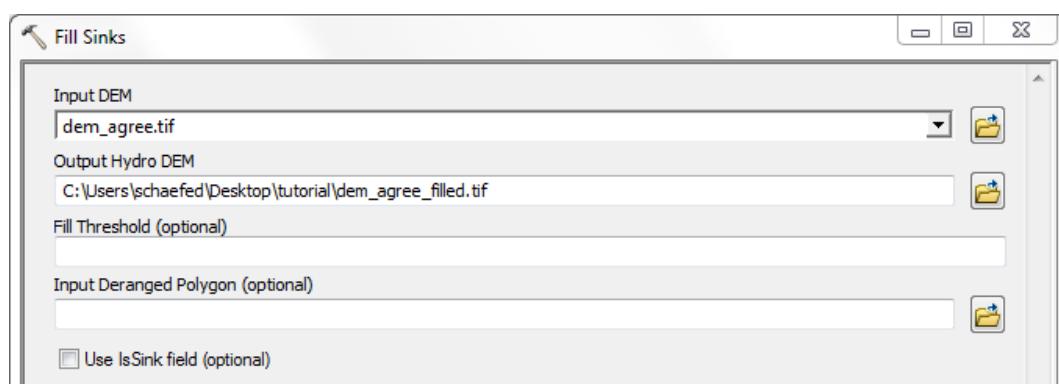


Figure 3.12: System Toolboxes -> Arc Hydro Tools -> Terrain Preprocessing -> Fill Sinks

- Calculate Flow Direction

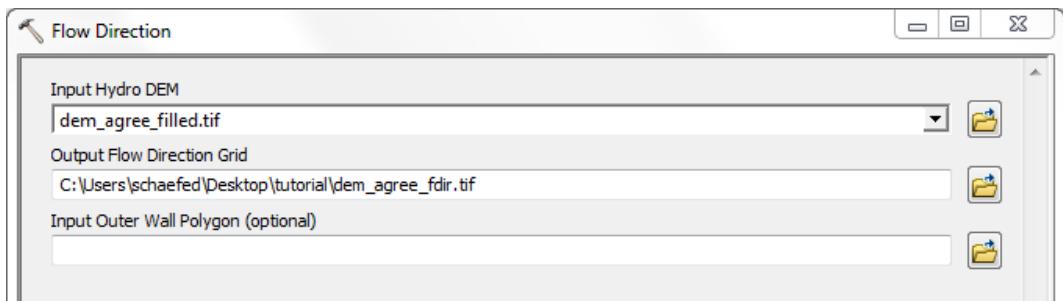


Figure 3.13: System Toolboxes -> Arc Hydro Tools -> Terrain Preprocessing -> Flow Direction

- Create a Flow Accumulation map

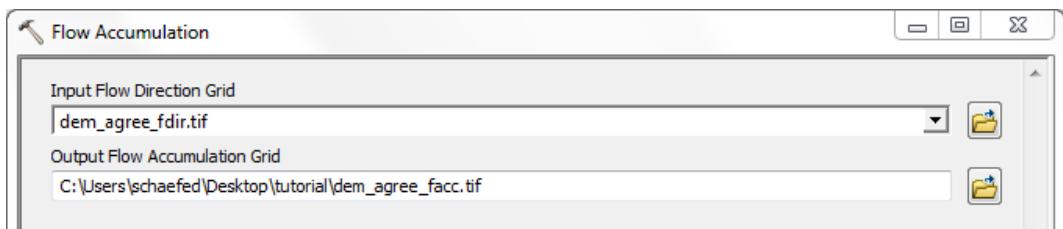


Figure 3.14: System Toolboxes -> Arc Hydro Tools -> Terrain Preprocessing -> Flow Accumulation

3.5.6 Gauges map

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

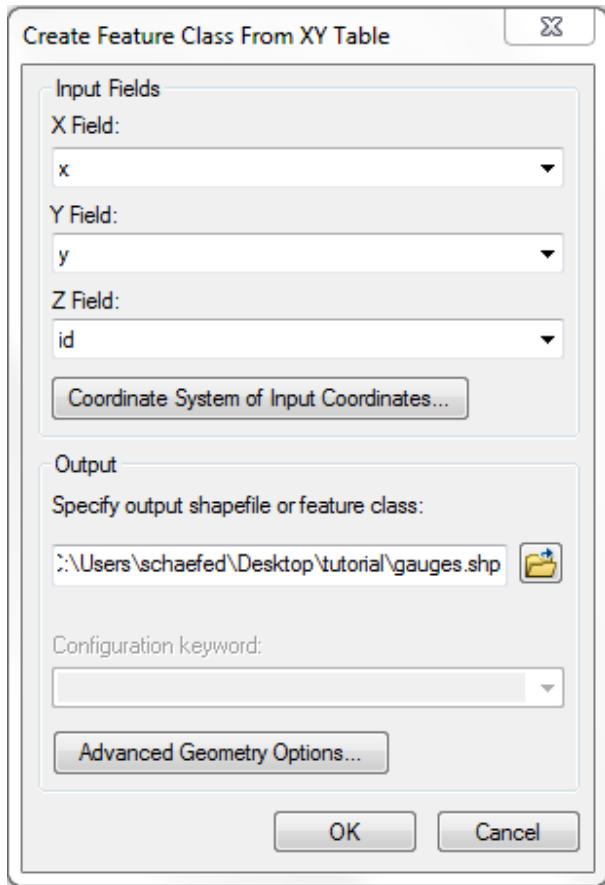


Figure 3.15: Right click on the table in Arc Catalog -> Create Feature Class -> From XY Table

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

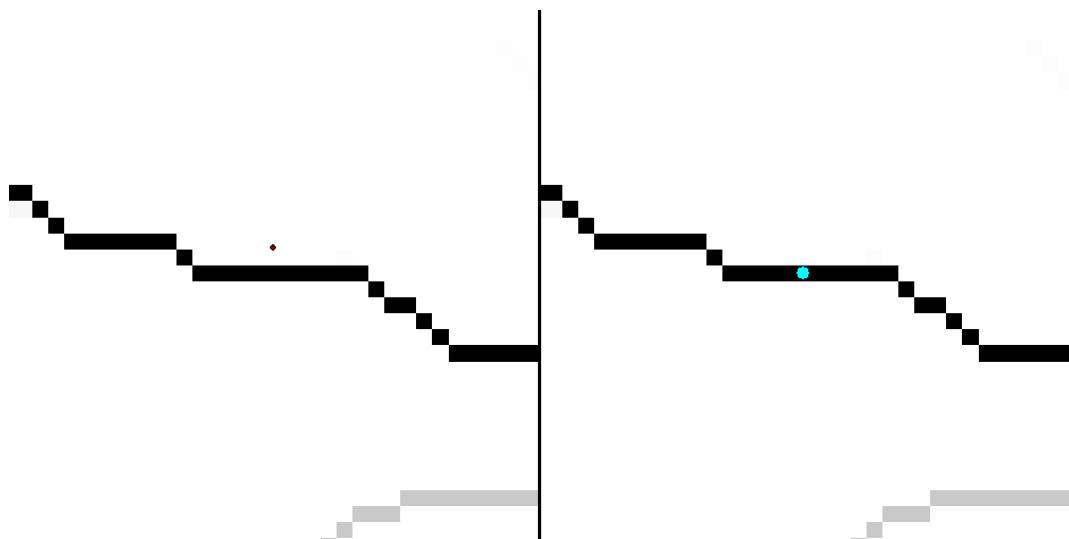


Figure 3.16: Mismatching gauges (left) should be corrected (right)

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

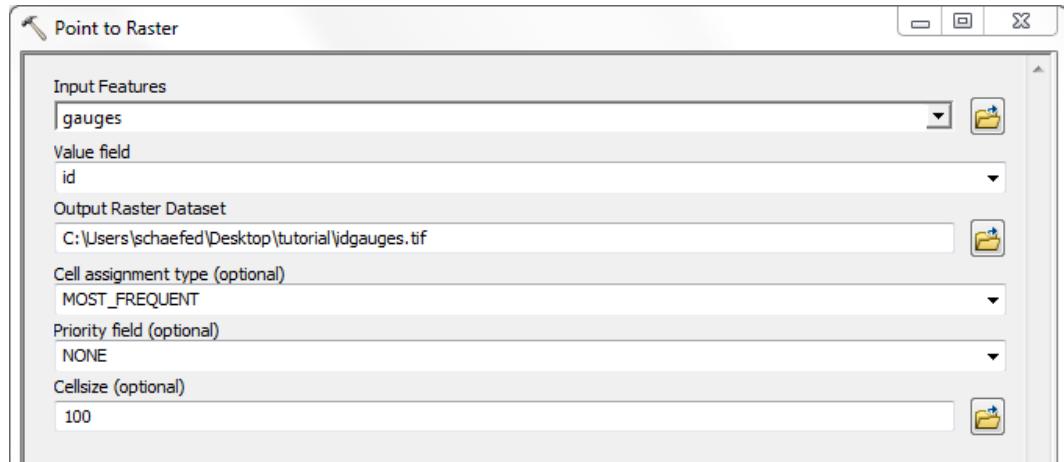


Figure 3.17: System Toolboxes -> Conversion Tools -> To Raster -> Point to Raster

3.5.7 Watershed delineation

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

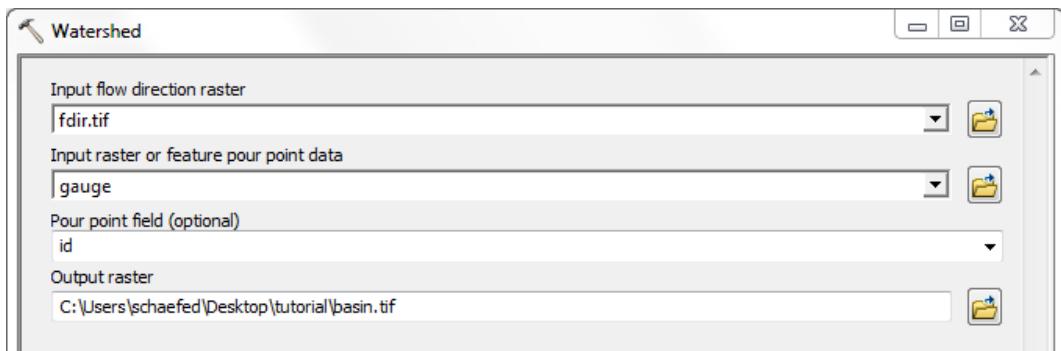


Figure 3.18: System Toolboxes -> Spatial Analyst -> Hydrology -> Watershed

3.5.8 Mask the datasets

Mask all mHM input raster files with the created watershed mask

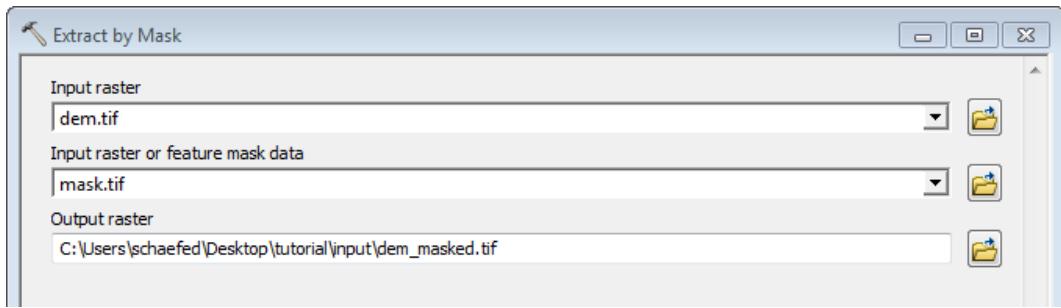


Figure 3.19: System Toolboxes -> Spatial Analysis Tools > Extraction > Extract by Mask

3.5.9 Write the ascii grids

Convert the processed and masked raster maps into ASCII files

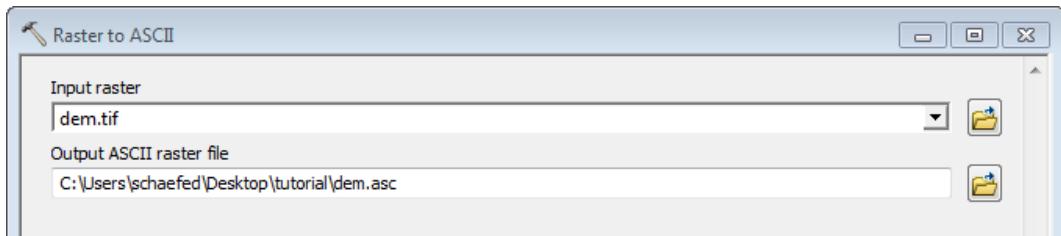


Figure 3.20: System Toolboxes -> Conversion Tools -> From Raster -> Raster to ASCII

3.6 Land Cover Data

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

Class	Description
1	Forest

Class	Description
2	Impervious
3	Pervious

3.7 Table Data

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

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

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

3.7.3 The LAI look-up table

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

Figure 3.23: A possible 'LAI_classdefinition.txt' file

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

3.7.4 The gauge files

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

Figure 3.24: A possible gauge file

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

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

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

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

```
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 previous 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° counter-clockwise rotation. Hence the values depicted NCVIEW using 8-flow direction convention (1,2,4,8,16,32,64,128) (i.e., 1 pointing to the east and then moving clockwise every 45°) 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 description 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 $\ln(\text{NSE})$ 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 the 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 `mhmm_parameters.nml` with `FinalParams.nml`, then run mHM in forward 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 m m s^{-1} for $\text{kg m}^{-2}\text{s}^{-1}$.
- 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 maximum 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 km², 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)

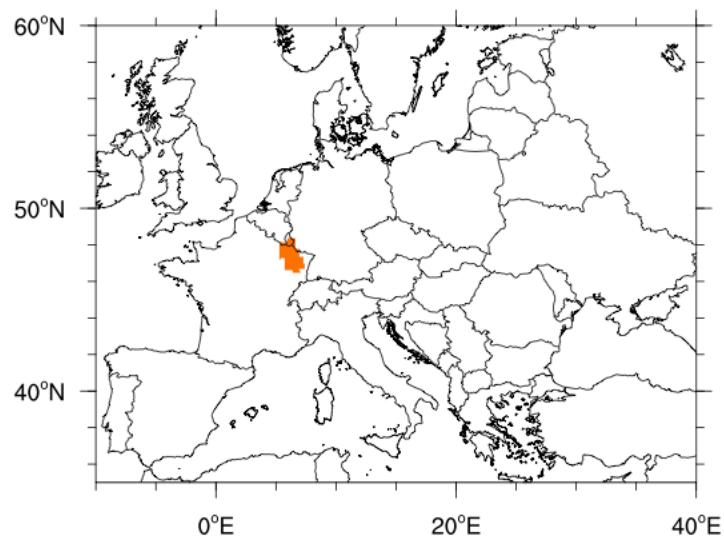


Figure 7.1: Location of the Moselle River basin upstream of Perl within the European domain

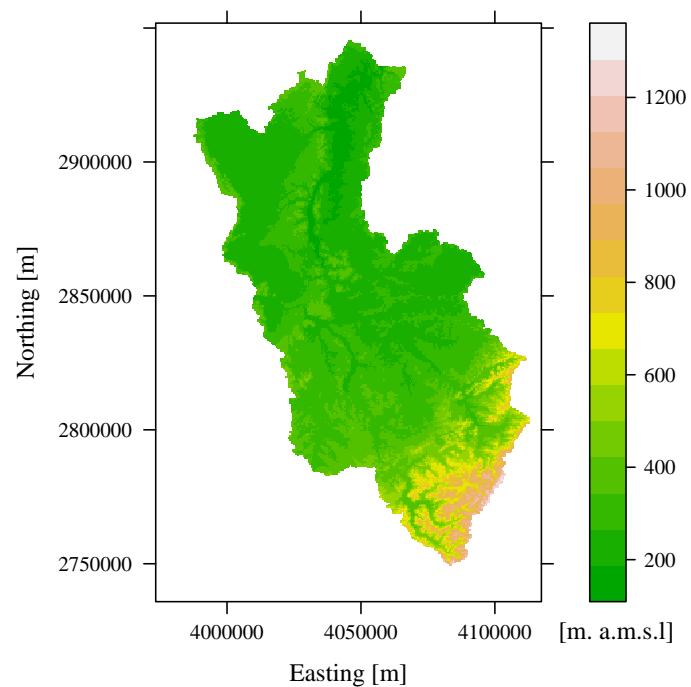


Figure 7.2: Digital elevation model for the Moselle River basin upstream of Perl

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

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 10

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 (SPAEOF): multiple-component evaluation of spatial patterns for optimization of hydrological models, *Geosci. Model Dev.*, 11, 1873-1886, <https://doi.org/10.5194/gmd-11-1873-2018/>
- Höllering, S., Wienhöfer, J., Ihringer, J., Samaniego, L., Zehe, E. (2018): Regional analysis of parameter sensitivity for simulation of streamflow and hydrological fingerprints, *Hydrol. Earth Syst. Sci.* 22 (1), 203 - 220, <https://www.hydrol-earth-syst-sci.net/22/203/2018/>
- Jing, M., Heße, F., Kumar, R., Wang, W., Fischer, T., Walther, M., Zink, M., Zech, A., Samaniego, L., Kolditz, O., Attinger, S. (2018): Improved regional-scale groundwater representation by the coupling of the mesoscale Hydrologic Model (mHM v5.7) to the groundwater model OpenGeoSys (OGS), *Geosci. Model Dev.* 11 (5), 1989 - 2007, <https://www.geosci-model-dev.net/11/1989/2018/>
- Peichl, M., Thober, S., Meyer, V., Samaniego, L. (2018): The effect of soil moisture anomalies on maize yield in Germany, *Nat. Hazards Earth Syst. Sci.* 18 (3), 889 - 906, <https://www.nat-hazards-earth-syst-sci.net/18/889/2018/>
- Samaniego, L., Thober, S., Kumar, R., Wanders, N., Rakovec, O., Pan, M., Zink, M., Sheffield, J., Wood, E.F., Marx, A. (2018): Anthropogenic warming exacerbates European soil moisture droughts, *Nat. Clim. Chang.* 8 (5), 421 - 426, <https://www.nature.com/articles/s41558-018-0138-5>
- Marx, A., Kumar, R., Thober, S., Rakovec, O., Wanders, N., Zink, M., Wood, E.F., Pan, M., Sheffield, J., Samaniego, L. (2018): Climate change alters low flows in Europe under global warming of 1.5, 2, and 3 degC, *Hydrol. Earth Syst. Sci.* 22 (2), 1017 - 1032, <https://www.hydrol-earth-syst-sci.net/22/1017/2018/>
- Schrön, M., Rosolem, R., Köhli, M., Piussi, L., Schröter, I., Iwema, J., Kögler, S., Oswald, S.E., Wollschläger, U., Samaniego, L., Dietrich, P., Zacharias, S. (2018): Cosmic-ray neutron rover surveys of field soil moisture and the influence of roads, *Water Resour. Res.*, <https://doi.org/10.1029/2017WR021719>
- Thober, S., Kumar, R., Wanders, N., Marx, A., Pan, M., Rakovec, O., Samaniego, L., Sheffield, J., Wood, E.F., Zink, M. (2018): Multi-model ensemble projections of European river floods and high flows at 1.5, 2, and 3 degrees global warming, *Environ. Res. Lett.* 13 (1), art. 014003, <http://iopscience.iop.org/article/10.1088/1748-9326/aa9e35>
- Zink, M., Mai, J., Cuntz, M., Samaniego, L. (2018): Conditioning a hydrologic model using patterns of remotely sensed land surface temperature, *Water Resour. Res.* 54 (4), 2976 - 2998, <https://doi.org/10.1002/2017WR021346>

- Samaniego, L., Kumar, R., Thober, S., Rakovec, O., Zink, M., Wanders, N., Eisner, S., Müller Schmied, H., Sutanudjaja, E.H., Warrach-Sagi, K., Attinger, S., (2017): Toward seamless hydrologic predictions across spatial scales, *Hydrol. Earth Syst. Sci.* 21 (9), 4323 - 4346, <https://www.hydrol-earth-syst-sci.net/21/4323/2017>
- Samaniego, L., Kumar, R., Breuer, L., Chamorro, A., Flörke, M., Pechlivanidis, I.G., Schäfer, D., Shah, H., Vetter, T., Wortmann, M., Zeng, X., (2017): Propagation of forcing and model uncertainties on to hydrological drought characteristics in a multi-model century-long experiment in large river basins, *Clim. Change* 141 (3), 435 - 449, <https://link.springer.com/article/10.1007/s10584-016-1778-y>
- Vigiak, O., Lutz, S., Mentafo, A., Chiogna, G., Ye, T., Majone, B., Beck, H., de Roo, A., Malagó, A., Bouraoui, F., Kumar, R., Samaniego, L., Merz, R., Gamvroudis, C., Skoulikidis, N., Nikolaidis, N.P., Bellin, A., Acuña, V., Mori, N., Ludwig, R., Pistocchi, A., (2018): Uncertainty of modelled flow regime for flow-ecological assessment in Southern Europe, *Sci. Total Environ.* 615 , 1028 - 1047, <https://www.sciencedirect.com/science/article/pii/S0048969717326475>
- Eisner, S., Flörke, M., Chamorro, A., Daggupati, P., Donnelly, C., Huang, J., Hundecha, Y., Koch, H., Kalugin, A., Krylenko, I., Mishra, V., Piniewski, M., Samaniego, L., Seidou, O., Wallner, M., Krysanova, V., (2017): An ensemble analysis of climate change impacts on streamflow seasonality across 11 large river basins, *Clim. Change* 141 (3), 401 - 417, <https://link.springer.com/article/10.1007/s10584-016-1844-5>
- Hattermann, F.F., Krysanova, V., Gosling, S.N., Dankers, R., Daggupati, P., Donnelly, C., Flörke, M., Huang, S., Motovilov, Y., Buda, S., Yang, T., Müller, C., Leng, G., Tang, Q., Portmann, F.T., Hagemann, S., Gerten, D., Wada, Y., Masaki, Y., Alemayehu, T., Satoh, Y., Samaniego, L., (2017): Cross-scale intercomparison of climate change impacts simulated by regional and global hydrological models in eleven large river basins, *Clim. Change* 141 (3), 561 - 576, <https://link.springer.com/article/10.1007/s10584-016-1829-4>
- Hattermann, F.F., Vetter, T., Breuer, L., Su, B., Daggupati, P., Donnelly, C., Fekete, B., Flörke, F., Gosling, S.N., Hoffmann, P., Liersch, S., Masaki, Y., Motovilov, Y., Müller, C., Samaniego, L., Stacke, T., Wada, Y., Yang, T., Krysanova, V., (2017): Sources of uncertainty in hydrological climate impact assessment: a cross-scale study, *Environ. Res. Lett.*, <http://iopscience.iop.org/article/10.1088/1748-9326/aa9938/meta>
- Mishra, V., Kumar, R., Shah, H.L., Samaniego, L., Eisner, S., Yang, T., (2017): Multimodel assessment of sensitivity and uncertainty of evapotranspiration and a proxy for available water resources under climate change, *Clim. Change* 141 (3), 451 - 465, <https://link.springer.com/article/10.1007/s10584-016-1886-8>
- Zink, M., Kumar, R., Cuntz, M., & Samaniego, L. (2017). A high-resolution dataset of water fluxes and states for Germany accounting for parametric uncertainty. *Hydrology and Earth System Sciences*, 21(3), 1769–1790. doi:10.5194/hess-21-1769-2017, <http://www.hydrol-earth-syst-sci.net/21/1769/2017/hess-21-1769-2017.html>
- Heße, F., Zink, M., Kumar, R., Samaniego, L., & Attinger, S. (2017). Spatially distributed characterization of soil-moisture dynamics using travel-time distributions. *Hydrology and Earth System Sciences*, 21(1), 549–570. doi:10.5194/hess-21-549-2017, <http://www.hydrol-earth-syst-sci.net/21/549/2017/>
- Baroni, G., Zink, M., Kumar, R., Samaniego, L., & Attinger, S. (2017). Effects of uncertainty in soil properties on simulated hydrological states and fluxes at different spatio-temporal scales. *Hydrology and Earth System Sciences*, 21(5), 2301–2320. doi:10.5194/hess-21-2301-2017, <http://www.hydrol-earth-syst-sci.net/21/2301/2017/hess-21-2301-2017.html>
- Wollschlaeger, U., Attinger, S., Borchardt, D., Brauns, M., Cuntz, M., Dietrich, P., ... Zacharias, S. (2017). The Bode hydrological observatory: a platform for integrated, interdisciplinary hydro-ecological research within the TERENO Harz/Central German Lowland Observatory. *Environmental Earth Sciences*, 76(1), 29. doi:10.1007/s12665-016-6327-5, <https://link.springer.com/article/10.1007/s12665-016-6327-5>

- Mueller, C., Zink, M., Samaniego, L., Krieg, R., Merz, R., Rode, M., & Knoeller, K. (2016). Discharge Driven Nitrogen Dynamics in a Mesoscale River Basin As Constrained by Stable Isotope Patterns. *Environmental Science & Technology*, 50(17), 9187–9196. doi:10.1021/acs.est.6b01057, <http://pubs.acs.org/doi/abs/10.1021/acs.est.6b01057>
- Zink, M., Samaniego, L., Kumar, R., Thober, S., Mai, J., Schaefer, D., & Marx, A. (2016). The German drought monitor. *Environmental Research Letters*, 11(7), 74002. doi:10.1088/1748-9326/11/7/074002, <http://iopscience.iop.org/article/10.1088/1748-9326/11/7/074002/meta>
- Marx, A., Samaniego, L., Kumar, R., Thober, S., Mai, J., & Zink, M. (2016). Der Duerremonitor - Aktuelle Information zur Bodenfeuchte in Deutschland. In *Wasserressourcen - Wissen in Flussgebieten vernetzen* (pp. 131–142). Forum fuer Hydrologie und Wasserbewirtschaftung, <http://www.ufz.de/index.php?en=20939&ufzPublicationIdentifier=17277>
- Rakovec, O., Kumar, R., Mai, J., Cuntz, M., Thober, S., Zink, M., Attinger, S., Schaefer, D., Schroen, M., Samaniego, L. (2016). Multiscale and Multivariate Evaluation of Water Fluxes and States over European River Basins. *Journal of Hydrometeorology*, 17(1), 287–307. doi:10.1175/JHM-D-15-0054.1, <http://journals.ametsoc.org/doi/abs/10.1175/JHM-D-15-0054.1>
- Rakovec, O., Kumar, R., Attinger, S. and Samaniego, L. (2016). Improving the realism of hydrologic model functioning through multivariate parameter estimation. *Water Resources Research*, 52. doi:10.1002/2016WR019430, <http://onlinelibrary.wiley.com/doi/10.1002/2016WR019430/full>
- Nijzink, R. C., Samaniego, L., Mai, J., Kumar, R., Thober, S., Zink, M., Schaefer, D., Savenije, H. H. G., and Hrachowitz, M.: The importance of topography-controlled sub-grid process heterogeneity and semi-quantitative prior constraints in distributed hydrological models, *Hydrol. Earth Syst. Sci.*, 20, 1151-1176, doi:10.5194/hess-20-1151-2016, 2016., <http://www.hydrol-earth-syst-sci.net/20/1151/2016/>
- Thober, S., Kumar, R., Sheffield, J., Mai, J., Schaefer, D., & Samaniego, L. (2015). Seasonal Soil Moisture Drought Prediction over Europe Using the North American Multi-Model Ensemble (NMME). *Journal of Hydrometeorology*, 16(6), 2329–2344. doi:10.1175/JHM-D-15-0053.1, <http://journals.ametsoc.org/doi/abs/10.1175/JHM-D-15-0053.1>
- Cuntz, M., Mai, J., Zink, M., Thober, S., Kumar, R., Schaefer, D., ... Samaniego, L. (2015). Computationally inexpensive identification of noninformative model parameters by sequential screening. *Water Resources Research*, 51(8), 6417–6441. doi:10.1002/2015WR016907, <http://onlinelibrary.wiley.com/doi/10.1002/2015WR016907/full>
- Livneh, B., Kumar, R., & Samaniego, L. (2015). Influence of soil textural properties on hydrologic fluxes in the Mississippi river basin. *Hydrological Processes*, 29(21), 4638-4655, <http://onlinelibrary.wiley.com/doi/10.1002/hyp.10601/full>
- Schoeniger, A., Woehling, T., Samaniego, L., & Nowak, W. (2014). Model selection on solid ground: Rigorous comparison of nine ways to evaluate Bayesian model evidence. *Water resources research*, 50(12), 9484-9513, <http://onlinelibrary.wiley.com/doi/10.1002/2014WR016062/full>
- Woehling, T., Samaniego, L., & Kumar, R. (2013). Evaluating multiple performance criteria to calibrate the distributed hydrological model of the upper Neckar catchment. *Environmental Earth Sciences*, 69(2), 453–468. doi:10.1007/s12665-013-2306-2, <https://link.springer.com/article/10.1007/s12665-013-2306-2>
- Samaniego, L., Kumar, R., & Zink, M. (2013). Implications of Parameter Uncertainty on Soil Moisture Drought Analysis in Germany. *Journal of Hydrometeorology*, 14(1), 47–68. doi:10.1175/JHM-D-12-075.1, <http://journals.ametsoc.org/doi/abs/10.1175/JHM-D-12-075.1>
- Kumar, R., Livneh, B., & Samaniego, L. (2013). Toward computationally efficient large-scale hydrologic predictions with a multiscale regionalization scheme. *Water Resources Research*, 49(9), 5700–5714. doi:10.1029/2012WR012431, <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 11

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

Bugs resolved from release 5.8:

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

Known bugs:

None.

Restrictions:

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

mHM v5.8 (Dec 2017)

New Features:

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

Bugs resolved from release 5.7:

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

Known bugs:

None.

Restrictions:

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

mHM v5.7 (Jun 2017)

New Features:

- New process descriptions for the soil evapotranspiration module:
 - Field capacity dependency to root fraction coefficient (`processCase (3) = 2`) – implemented by [G-EUS](#).
 - Jarvis (1989, J. Hydrol.) evapotranspiration reduction (`processCase (3) = 3`) – implemented by [G-EUS](#).
- Use local, monthly LAI climatology instead of look-up table, i.e., `LAI_classdefinition.txt` (`timeStep_LAI_input=1`).
- New objective functions for model calibration
 - Calibration of mHM using catchment average evapotranspiration (`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 now (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 12

Modules Index

12.1 Modules List

Here is a list of all modules with brief descriptions:

dummy_mpr	75
dummy_mrm	75
mo_anneal	75
mo_append Append values on existing arrays	79
mo_canopy_interc Canopy interception	87
mo_common_constants Provides constants commonly used by mHM, mRM and MPR	89
mo_common_file Provides file names and units for mRM	92
mo_common_functions Provides small utility functions used by multiple parts of the code (mHM, mRM, MPR)	94
mo_common_mhm_mrm_file Provides file names and units for mHM	95
mo_common_mhm_mrm_read_config Reading of main model configurations	96
mo_common_mhm_mrm_variables Provides structures needed by mHM, mRM and/or mpr	100
mo_common_read_config Reading of main model configurations	106
mo_common_read_data TODO: add description	108
mo_common_restart TODO: add description	111
mo_common_variables Provides structures needed by mHM, mRM and/or mpr	114
mo_constants Provides computational, mathematical, physical, and file constants	121
mo_corr	131
mo_dds Dynamically Dimensioned Search (DDS)	138
mo_errormeasures	142
mo_file Provides file names and units for mHM	167
mo_finish Finish a program gracefully	169

mo_global_variables	Global variables ONLY used in reading, writing and startup	171
mo_grid	TODO: add description	184
mo_init_states	Initialization of all state variables of mHM	190
mo_julian	Julian date conversion routines	192
mo_kind	Define number representations	221
mo_linfit	Fitting a straight line	224
mo_mcmc	This module is Monte Carlo Markov Chain sampling of a posterior parameter distribution	225
mo_message	Write out concatenated strings	228
mo_meteo_forcings	Prepare meteorological forcings data for mHM	230
mo_mhm	Call all main processes of mHM	238
mo_mhm_constants	Provides mHM specific constants	244
mo_mhm_eval	Runs mhm with a specific parameter set and returns required variables, e.g. runoff	249
mo_mhm_read_config	Reading of main model configurations	253
mo_moment	256
mo_mpr_constants	Provides MPR specific constants	261
mo_mpr_eval	Runs MPR and writes to global effective parameters	268
mo_mpr_file	Provides file names and units for mRM	271
mo_mpr_global_variables	Global variables for mpr only	276
mo_mpr_pet	TODO: add description	286
mo_mpr_read_config	Read mpr config	291
mo_mpr_restart	Reading and writing states, fluxes and configuration for restart of mHM	293
mo_mpr_runoff	Multiscale parameter regionalization for runoff generation	297
mo_mpr_smhorizons	Setting up the soil moisture horizons	300
mo_mpr_soilmoist	Multiscale parameter regionalization (MPR) for soil moisture	302
mo_mpr_startup	Startup procedures for mHM	309
mo_mrm_constants	Provides mRM specific constants	314
mo_mrm_eval	Runs mrm with a specific parameter set and returns required variables, e.g. runoff	316
mo_mrm_file	Provides file names and units for mRM	318
mo_mrm_global_variables	Global variables for mRM only	323

mo_mrm_init	Wrapper for initializing Routing	333
mo_mrm_mpr	Perform Multiscale Parameter Regionalization on Routing Parameters	343
mo_mrm_net_startup	Startup drainage network for mHM	345
mo_mrm_objective_function_runoff	Objective Functions for Optimization of mHM/mRM against runoff	359
mo_mrm_read_config	Read mRM config	385
mo_mrm_read_data	This module contains all routines to read mRM data from file	389
mo_mrm_restart	Restart routines	393
mo_mrm_routing	Performs runoff routing for mHM at level L11	397
mo_mrm_signatures	Module with calculations for several hydrological signatures	405
mo_mrm_write	Write of discharge and restart files	414
mo_mrm_write_fluxes_states	Creates NetCDF output for different fluxes and state variables of mHM	424
mo_multi_param_reg	Multiscale parameter regionalization (MPR)	433
mo_ncread		445
mo_ncwrite		465
mo_netcdf	NetCDF Fortran 90 interface wrapper	493
mo_neutrons	Models to predict neutron intensities above soils	557
mo_nml	Deal with namelist files	566
mo_objective_function	Objective Functions for Optimization of mHM	572
mo_optimization	Wrapper subroutine for optimization against runoff and sm	588
mo_optimization_utils		590
mo_orderpack	Sort and ranking routines	591
mo_percentile		608
mo_pet	Module for calculating reference/potential evapotranspiration [mm s-1]	610
mo_prepare_gridded_lai	Prepare daily LAI fields (e.g., MODIS data) for mHM	618
mo_read_forcing_nc	Reads forcing input data	621
mo_read_latlon	Reading latitude and longitude coordinates for each basin	626
mo_read_lut	Routines reading lookup tables (lut)	628
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_read_wrapper	Wrapper for all reading routines	641

mo_restart	Reading and writing states, fluxes and configuration for restart of mHM	644
mo_runoff	Runoff generation for the unsaturated zone, saturated zone (or groundwater zone), and runoff accumulation	648
mo_sce	Shuffled Complex Evolution optimization algorithm	652
mo_set_netcdf_outputs	Defines the structure of the netCDF to write the output in	660
mo_snow_accum_melt	Snow melting and accumulation	661
mo_soil_database	Generating soil database from input file	662
mo_soil_moisture	Soil moisture of the different layers	665
mo_spatial_agg_disagg_forcing	Spatial aggregation or disaggregation of meteorological input data	669
mo_spatialsimilarity	Routines for bias insensitive comparison of spatial patterns	671
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	672
mo_startup	Startup procedures for mHM	674
mo_string_utils	String utilities	678
mo_template	Template for future module developments	685
mo_temporal_aggregation	Temporal aggregation for time series (averaging)	687
mo_temporal_disagg_forcing	Temporal disaggregation of daily input values	688
mo_timer	Timing routines	690
mo_upscaling_operators	Module containing upscaling operators	699
mo_utils	General utilities for the CHS library	706
mo_write_ascii	Module to write ascii file output	711
mo_write_fluxes_states	Creates NetCDF output for different fluxes and state variables of mHM	715
mo_xor4096		724

Chapter 13

Data Type Index

13.1 Data Types List

Here are the data types with brief descriptions:

mo_moment::absdev	729
mo_anneal::anneal	
Anneal	729
mo_append::append	
Append (rows) scalars, vectors, and matrixes onto existing array	732
mo_corr::arth	737
mo_ncwrite::attribute	739
mo_corr::autocoeffk	740
mo_corr::autocorr	741
mo_moment::average	741
mo_mrm_global_variables::basininfo_mrm	742
mo_errormeasures::bias	744
mo_moment::central_moment	746
mo_moment::central_moment_var	746
mo_standard_score::classified_standard_score	
Calculates the classified standard score (e.g. classes are months)	747
mo_corr::corr	748
mo_moment::correlation	749
mo_moment::covariance	749
mo_corr::crosscoeffk	750
mo_corr::crosscorr	751
mo_orderpack::ctrper	751
mo_temporal_aggregation::day2mon_average	
Day-to-month average (day2mon_average)	752
mo_ncwrite::dims	753
mo_ncwrite::dump_netcdf	754
mo_utils::eq	758
mo_utils::equal	
Comparison of real values	758
mo_optimization_utils::eval_interface	759
mo_orderpack::fndnth	760
mo_corr::four1	761
mo_corr::fourrow	761
mo_mrm_global_variables::gaugingstation	762
mo_utils::ge	763
mo_anneal::generate_neighborhood_weight	763
mo_ncread::get_ncvar	764
mo_xor4096::get_timeseed	769

mo_anneal::gettemperature	770
GetTemperature	770
mo_utils::greaterequal	772
mo_common_variables::grid	773
mo_common_variables::gridremapper	776
mo_temporal_aggregation::hour2day_average	777
Hour-to-day average (hour2day_average)	777
mo_orderpack::indmed	778
mo_orderpack::indnth	779
mo_orderpack::inspar	780
mo_orderpack::inssor	781
mo_utils::is_finite	781
.true. if not IEEE Inf, IEEE NaN, nor IEEE Inf nor IEEE NaN, respectively	781
mo_utils::is_nan	782
mo_utils::is_normal	783
mo_errormeasures::kge	783
Kling-Gupta-Efficiency measure	783
mo_errormeasures::kgenocorr	785
Kling-Gupta-Efficiency measure without correlation	785
mo_moment::kurtosis	787
mo_utils::le	788
mo_utils::lesserequal	789
mo_linfit::linfit	789
Fits a straight line to input data by minimizing chi^2	789
mo_errormeasures::lnnse	790
mo_utils::locate	792
Find closest values in a monotonic series, returns the indexes	792
mo_errormeasures::mae	793
mo_mcmc::mcmc	794
This module is Monte Carlo Markov Chain sampling of a posterior parameter distribution (measurement errors are either known or modeled)	794
mo_mcmc::mcmc_stddev	798
This module is Monte Carlo Markov Chain sampling of a posterior parameter distribution (measurement errors are neither known nor modeled)	798
mo_template::mean	802
The average	802
mo_moment::mean	804
mo_percentile::median	804
mo_moment::mixed_central_moment	805
mo_moment::mixed_central_moment_var	805
mo_moment::moment	806
mo_orderpack::mrgref	807
mo_orderpack::mrgrnk	807
mo_errormeasures::mse	808
mo_orderpack::mulcnt	810
mo_percentile::n_element	810
mo_netcdf::ncdataset	811
Provides basic file modification functionality	811
mo_netcdf::ncdimension	822
Provides the dimension access functionality	822
mo_netcdf::ncvariable	842
mo_utils::ne	842
mo_orderpack::nearless	843
mo_spatialsimilarity::nndv	843
Calculates the number of neighboring dominating values, a measure for spatial dissimilarity	843
mo_utils::notequal	845
mo_errormeasures::nse	846

mo_string_utils::num2str	Convert to string	847
mo_string_utils::numarray2str	Convert to string	849
mo_optimization_utils::objective_interface		850
mo_orderpack::omedian		850
mo_mrm_write_fluxes_states::outputdataset		851
mo_write_fluxes_states::outputdataset		855
mo_write_fluxes_states::outputvariable		859
mo_mrm_write_fluxes_states::outputvariable		864
mo_append::paste	Paste (columns) scalars, vectors, and matrixes onto existing array	868
mo_spatialsimilarity::pd	Calculates pattern dissimilarity (PD) measure	872
mo_percentile::percentile		874
mo_common_variables::period		875
mo_percentile::qmedian		877
mo_orderpack::rapknn		877
mo_read_spatial_data::read_spatial_data_ascii	Reads spatial data files of ASCII format	878
mo_corr::realfft		880
mo_orderpack::refpar		881
mo_orderpack::refsor		882
mo_orderpack::rinpar		882
mo_errormeasures::rmse		883
mo_orderpack::rnkpar		884
mo_errormeasures::sae		885
mo_julian::setcalendar		886
mo_moment::skewness		887
mo_mpr_global_variables::soiltype		888
mo_orderpack::sort	Unconditional ranking	891
mo_orderpack::sort_index		894
mo_spatial_agg_disagg_forcing::spatial_aggregation	Spatial aggregation of meterological variables	895
mo_spatial_agg_disagg_forcing::spatial_disaggregation	Spatial disaggregation of meterological variables	896
mo_utils::special_value	Special IEEE values	897
mo_kind::sprs2_dp	Double Precision Numerical Recipes types for sparse arrays	899
mo_kind::sprs2_sp	Single Precision Numerical Recipes types for sparse arrays	900
mo_errormeasures::sse		901
mo_standard_score::standard_score	Calculates the standard score / normalization (anomaly) / z-score	903
mo_moment::stddev		904
mo_corr::swap		905
mo_utils::swap	Swap to values or two elements in array	905
mo_global_variables::twsstructure		907
mo_orderpack::uniinv		908
mo_orderpack::unipar		909
mo_orderpack::unirnk		909
mo_orderpack::unista		910
mo_mpr_restart::unpack_field_and_write		911
mo_utils::TODO	TODO: add description	

mo_restart::unpack_field_and_write	912
TODO: add description	912
mo_orderpack::valmed	914
mo_orderpack::valnth	915
mo_ncwrite::var2nc	916
Extended <code>dump_netcdf</code> for multiple variables	916
mo_ncwrite::variable	923
mo_moment::variance	928
mo_errormeasures::wnse	929
mo_xor4096::xor4096	930
mo_xor4096::xor4096g	932

Chapter 14

File Index

14.1 File List

Here is a list of all files with brief descriptions:

mhmm_driver.f90	933
mo_anneal.f90	936
mo_append.f90	937
mo_canopy_interc.f90	938
mo_common_constants.f90	938
mo_common_file.f90	939
mo_common_functions.f90	939
mo_common_mHM_mRM_file.f90	940
mo_common_mHM_mRM_read_config.f90	940
mo_common_mHM_mRM_variables.f90	940
mo_common_read_config.f90	941
mo_common_read_data.f90	941
mo_common_restart.f90	942
mo_common_variables.f90	942
mo_constants.f90	943
mo_corr.f90	945
mo_dds.f90	946
mo_errormeasures.f90	946
mo_file.f90	948
mo_finish.f90	949
mo_global_variables.f90	949
mo_grid.f90	951
mo_init_states.f90	951
mo_julian.f90	952
mo_kind.f90	953
mo_linfit.f90	954
mo_mcmc.f90	954
mo_message.f90	955
mo_meteo_forcings.f90	955
mo_mhm.f90	955
mo_mhm_constants.f90	956
mo_mhm_eval.f90	957
mo_mhm_read_config.f90	957
mo_moment.f90	957
mo_mpr_constants.f90	958
mo_mpr_eval.f90	959
mo_mpr_file.f90	960
mo_mpr_global_variables.f90	961

mo_mpr_pet.f90	962
mo_mpr_read_config.f90	963
mo_mpr_restart.f90	963
mo_mpr_runoff.f90	963
mo_mpr_smhorizons.f90	964
mo_mpr_soilmoist.f90	964
mo_mpr_startup.f90	964
mo_mrm_constants.f90	965
mo_mrm_eval.f90	965
mo_mrm_file.f90	965
mo_mrm_global_variables.f90	966
mo_mrm_init.f90	968
mo_mrm_mpr.f90	968
mo_mrm_net_startup.f90	969
mo_mrm_objective_function_runoff.f90	969
mo_mrm_read_config.f90	970
mo_mrm_read_data.f90	971
mo_mrm_restart.f90	971
mo_mrm_routing.f90	971
mo_mrm_signatures.f90	972
mo_mrm_write.f90	973
mo_mrm_write_fluxes_states.f90	973
mo_multi_param_reg.f90	974
mo_ncread.f90	975
mo_ncwrite.f90	976
mo_netcdf.f90	977
mo_neutrons.f90	980
mo_nml.f90	981
mo_objective_function.f90	981
mo_optimization.f90	982
mo_optimization_utils.f90	982
mo_orderpack.f90	983
mo_percentile.f90	985
mo_pet.f90	985
mo_prepare_gridded_lai.f90	986
mo_read_forcing_nc.f90	986
mo_read_latlon.f90	987
mo_read_lut.f90	987
mo_read_optional_data.f90	987
mo_read_spatial_data.f90	988
mo_read_timeseries.f90	988
mo_read_wrapper.f90	988
mo_restart.f90	989
mo_runoff.f90	989
mo_sce.f90	989
mo_set_netcdf_outputs.f90	992
mo_snow_accum_melt.f90	992
mo_soil_database.f90	992
mo_soil_moisture.f90	993
mo_spatial_agg_disagg_forcing.f90	993
mo_spatialsimilarity.f90	994
mo_standard_score.f90	994
mo_startup.f90	994
mo_string_utils.f90	995
mo_template.f90	996
mo_temporal_aggregation.f90	996
mo_temporal_disagg_forcing.f90	997
mo_timer.f90	997

mo_upscaling_operators.f90	998
mo_utils.f90	998
mo_write_ascii.f90	999
mo_write_fluxes_states.f90	1000
mo_xor4096.f90	1001
mpr_driver.f90	1001
mrm_driver.f90	1003

Chapter 15

Module Documentation

15.1 dummy_mpr Module Reference

15.2 dummy_mrm Module Reference

15.3 mo_anneal Module Reference

Data Types

- interface `anneal`
anneal
- interface `generate_neighborhood_weight`
- interface `gettemperature`
GetTemperature.

Functions/Subroutines

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

15.3.1 Function/Subroutine Documentation

15.3.1.1 anneal_dp()

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

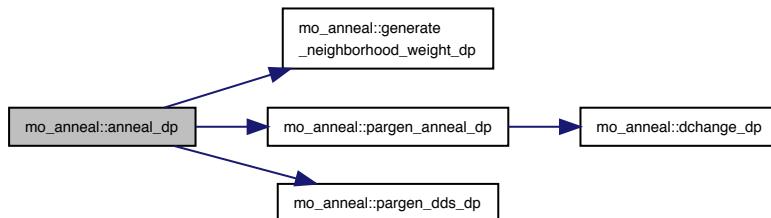
```

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

```

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

Here is the call graph for this function:



15.3.1.2 dchange_dp()

```

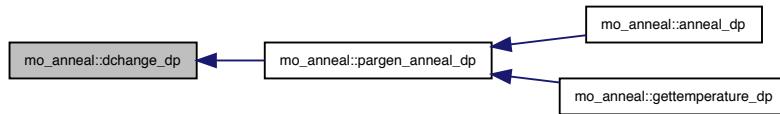
real(dp) function mo_anneal::dchange_dp (
    real(dp), intent(in) delta,
    integer(i4), intent(in) iDigit,
    integer(i4), intent(in) isZero ) [private]

```

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

Referenced by `pargen_anneal_dp()`.

Here is the caller graph for this function:



15.3.1.3 generate_neighborhood_weight_dp()

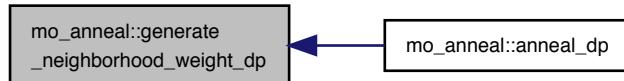
```

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

```

Referenced by anneal_dp().

Here is the caller graph for this function:



15.3.1.4 gettemperature_dp()

```

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

```

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

Here is the call graph for this function:



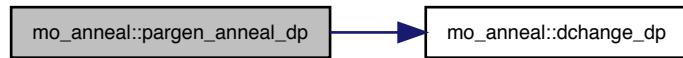
15.3.1.5 `pargen_anneal_dp()`

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

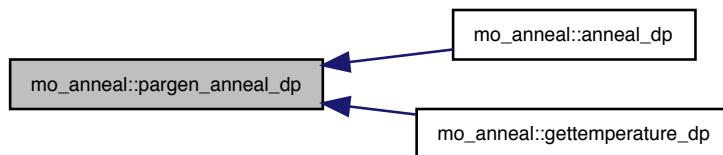
References `dchange_dp()`, `mo_kind::dp`, and `mo_kind::i4`.

Referenced by `anneal_dp()`, and `gettemperature_dp()`.

Here is the call graph for this function:



Here is the caller graph for this function:

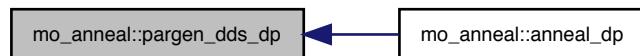


15.3.1.6 pargen_dds_dp()

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

Referenced by anneal_dp().

Here is the caller graph for this function:



15.4 mo_append Module Reference

Append values on existing arrays.

Data Types

- interface [append](#)
Append (rows) scalars, vectors, and matrixes onto existing array.
- interface [paste](#)
Paste (columns) scalars, vectors, and matrixes onto existing array.

Functions/Subroutines

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

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

15.4.1 Detailed Description

Append values on existing arrays.

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

Author

Juliane Mai

Date

Aug 2012

15.4.2 Function/Subroutine Documentation

15.4.2.1 `append_char_3d()`

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

15.4.2.2 append_char_m_m()

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

15.4.2.3 append_char_v_s()

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

15.4.2.4 append_char_v_v()

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

15.4.2.5 append_dp_3d()

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

15.4.2.6 append_dp_m_m()

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

15.4.2.7 append_dp_v_s()

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

15.4.2.8 append_dp_v_v()

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

15.4.2.9 `append_i4_3d()`

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

15.4.2.10 `append_i4_m_m()`

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

15.4.2.11 `append_i4_v_s()`

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

15.4.2.12 `append_i4_v_v()`

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

15.4.2.13 `append_i8_3d()`

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

15.4.2.14 `append_i8_m_m()`

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

15.4.2.15 append_i8_v_s()

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

15.4.2.16 append_i8_v_v()

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

15.4.2.17 append_lgt_3d()

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

15.4.2.18 append_lgt_m_m()

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

15.4.2.19 append_lgt_v_s()

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

15.4.2.20 append_lgt_v_v()

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

15.4.2.21 append_sp_3d()

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

15.4.2.22 `append_sp_m_m()`

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

15.4.2.23 `append_sp_v_s()`

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

15.4.2.24 `append_sp_v_v()`

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

15.4.2.25 `paste_char_m_m()`

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

15.4.2.26 `paste_char_m_s()`

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

15.4.2.27 `paste_char_m_v()`

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

15.4.2.28 `paste_dp_m_m()`

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

```

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

```

15.4.2.29 paste_dp_m_s()

```

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

```

15.4.2.30 paste_dp_m_v()

```

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

```

15.4.2.31 paste_i4_m_m()

```

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

```

15.4.2.32 paste_i4_m_s()

```

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

```

15.4.2.33 paste_i4_m_v()

```

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

```

15.4.2.34 paste_i8_m_m()

```

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

```

15.4.2.35 `paste_i8_m_s()`

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

15.4.2.36 `paste_i8_m_v()`

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

15.4.2.37 `paste_lgt_m_m()`

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

15.4.2.38 `paste_lgt_m_s()`

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

15.4.2.39 `paste_lgt_m_v()`

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

15.4.2.40 `paste_sp_m_m()`

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

15.4.2.41 `paste_sp_m_s()`

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

15.4.2.42 paste_sp_m_v()

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

15.5 mo_canopy_interc Module Reference

Canopy interception.

Functions/Subroutines

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

15.5.1 Detailed Description

Canopy interception.

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

Authors

Vladyslav Prykhodko

Date

Dec 2012

15.5.2 Function/Subroutine Documentation

15.5.2.1 canopy_interc()

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

Canopy interception.

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

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

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 intercepted by the vegetation.

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

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

Parameters

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

Authors

Vladyslav Prykhodko

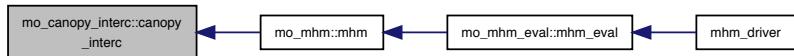
Date

Dec 2012

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

Referenced by mo_mhm::mhm().

Here is the caller graph for this function:



15.6 mo_common_constants Module Reference

Provides constants commonly used by mHM, mRM and MPR.

Variables

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

15.6.1 Detailed Description

Provides constants commonly used by mHM, mRM and MPR.

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

Authors

Robert Schwegelpe

Date

Dec 2017

15.6.2 Variable Documentation

15.6.2.1 dayhours

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

15.6.2.2 daysecs

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

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

15.6.2.3 eps_dp

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

epsilon(1.0) in double precision

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

15.6.2.4 eps_sp

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

epsilon(1.0) in single precision

15.6.2.5 hoursecs

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

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

15.6.2.6 maxnlcovers

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

Referenced by mo_common_read_config::common_read_config().

15.6.2.7 maxnobasins

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

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

15.6.2.8 ncolpars

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

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

15.6.2.9 nodata_dp

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

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

15.6.2.10 nodata_i4

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

Referenced by mo_soil_database::generate_soil_database(), mo_grid::init_lowres_level(), mo_multi_param::reg::karstic_layer(), mo_mrm_init::l0_check_input_routing(), mo_upscaling_operators::l0_fractionalcover_in_lx(), mo_mrm_net_startup::l11_flow_direction(), mo_mrm_net_startup::l11_l1_mapping(), mo_mrm_net_startup::l11_link_location(), mo_mrm_net_startup::l11_routing_order(), mo_mrm_net_startup::l11_set_drain_outlet_gauges(), mo_mrm_net_startup::l11_set_network_topology(), mo_mrm_net_startup::l11_stream_features(), mo_mhm::read_config::mhmm_read_config(), mo_multi_param_reg::mpr(), mo_mpr_runoff::mpr_runoff(), mo_mpr_soilmoist::mpr_sm(), mo_mrm_init::mrm_init(), mo_mrm_read_config::mrm_read_config(), mo_mrm_read_data::mrm::read_l0_data(), mo_mrm_restart::mrm_write_restart(), mo_read_wrapper::read_data(), mo_common_read_data::read_lcover(), mo_soil_database::read_soil_lut(), mo_mrm_read_data::rotate_fdir_variable(), mo_common::read_config::set_land_cover_scenes_id(), mo_mpr_restart::write_eff_params(), and mo_common_restart::write_grid_info().

15.6.2.11 p1_initstatefluxes

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

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

15.6.2.12 yeardays

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

Referenced by mo_pet::extraterr_rad_approx().

15.6.2.13 yearmonths

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

Referenced by mo_read_lut::read_lai_lut().

15.6.2.14 yearmonths_i4

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

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

15.7 mo_common_file Module Reference

Provides file names and units for mRM.

Variables

- character(len= *), parameter **file_dem** = 'dem.asc'

DEM input data file.

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

15.7.1 Detailed Description

Provides file names and units for mRM.

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

Authors

Matthias Cuntz, Stephan Thober

Date

Aug 2015

15.7.2 Variable Documentation

15.7.2.1 file_config

```
character(len=*), parameter mo_common_file::file_config = 'ConfigFile.log'  
file defining mHM's outputs
```

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

15.7.2.2 file_dem

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

Referenced by `mo_common_read_data::read_dem()`.

15.7.2.3 uconfig

```
integer, parameter mo_common_file::uconfig = 68  
Unit for file defining mHM's outputs.
```

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

15.7.2.4 udem

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

Unit for DEM input data file.

Referenced by mo_common_read_data::read_dem().

15.7.2.5 ulcoverclass

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

Unit for LCover input data file.

Referenced by mo_common_read_data::read_lcover().

15.8 mo_common_functions Module Reference

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

Functions/Subroutines

- logical function, public [in_bound](#) (params)

TODO: add description.

15.8.1 Detailed Description

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

Provides the functions in_bound used to check global_parameter ranges

Authors

Robert Scheweppe

Date

Dec 2017

15.8.2 Function/Subroutine Documentation

15.8.2.1 in_bound()

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

TODO: add description.

TODO: add description

Parameters

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

Authors

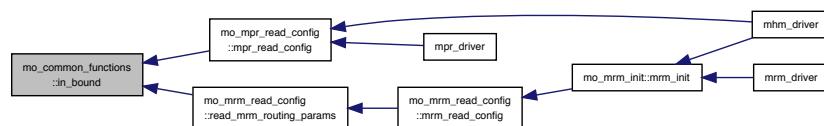
Robert Scheweppe

Date

Jun 2018

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

Here is the caller graph for this function:



15.9 mo_common_mhm_mrm_file Module Reference

Provides file names and units for mHM.

Variables

- character(len=*), parameter `file_opti` = 'FinalParam.out'
file defining optimization outputs (objective and parameter set)
- integer, parameter `uopti` = 72
Unit for file optimization outputs (objective and parameter set)
- character(len=*), parameter `file_opti_nml` = 'FinalParam.nml'
file defining optimization outputs in a namelist format (parameter set)
- integer, parameter `uopti_nml` = 73
Unit for file optimization outputs in a namelist format (parameter set)

15.9.1 Detailed Description

Provides file names and units for mHM.

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

Authors

Matthias Cuntz

Date

Jan 2012

15.9.2 Variable Documentation

15.9.2.1 file_opti

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

file defining optimization outputs (objective and parameter set)

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

15.9.2.2 file_opti_nml

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

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

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

15.9.2.3 uopti

integer, parameter mo_common_mhm_mrm_file::uopti = 72

Unit for file optimization outputs (objective and parameter set)

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

15.9.2.4 uopti_nml

integer, parameter mo_common_mhm_mrm_file::uopti_nml = 73

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

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

15.10 mo_common_mhm_mrm_read_config Module Reference

Reading of main model configurations.

Functions/Subroutines

- subroutine, public [common_mhm_mrm_read_config](#) (file_namelist, unamelist)
Read main configurations for common parts.
- subroutine, public [check_optimization_settings](#)
TODO: add description.
- subroutine, public [common_check_resolution](#) (do_message, allow_subgrid_routing)
TODO: add description.

15.10.1 Detailed Description

Reading of main model configurations.

This routine reads the configurations of common program parts

Authors

Matthias Zink

Date

Dec 2012

15.10.2 Function/Subroutine Documentation

15.10.2.1 check_optimization_settings()

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

TODO: add description.

TODO: add description

Authors

Robert Schwepppe

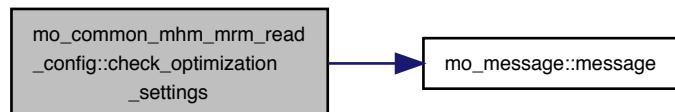
Date

Jun 2018

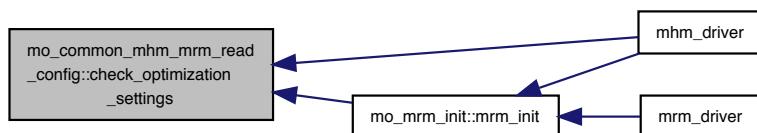
References mo_common_mhm_mrm_variables::dds_r, mo_common_variables::global_parameters, mo_message::message(), mo_common_mhm_mrm_variables::niterations, mo_common_mhm_mrm_variables::sce_ngs, mo_common_mhm_mrm_variables::sce_npg, and mo_common_mhm_mrm_variables::sce_nps.

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

Here is the call graph for this function:



Here is the caller graph for this function:



15.10.2.2 common_check_resolution()

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

TODO: add description.

TODO: add description

Parameters

in	<i>logical :: do_message</i>	
in	<i>logical :: allow_subgrid_routing</i>	

Authors

Robert Scheweppe

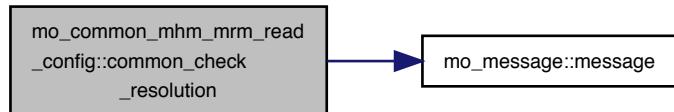
Date

Jun 2018

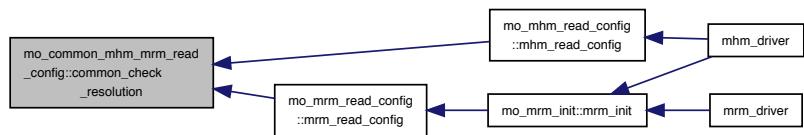
References `mo_message::message()`, `mo_common_variables::nbasins`, `mo_common_variables::resolutionhydrology`, and `mo_common_mhm_mrm_variables::resolutionrouting`.

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

Here is the call graph for this function:



Here is the caller graph for this function:



15.10.2.3 common_mhm_mrm_read_config()

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

Read main configurations for common parts.

TODO: add description

Parameters

in	character(*) :: <i>file_namelist</i>	
in	integer :: <i>unamelist</i>	

Authors

Matthias Zink

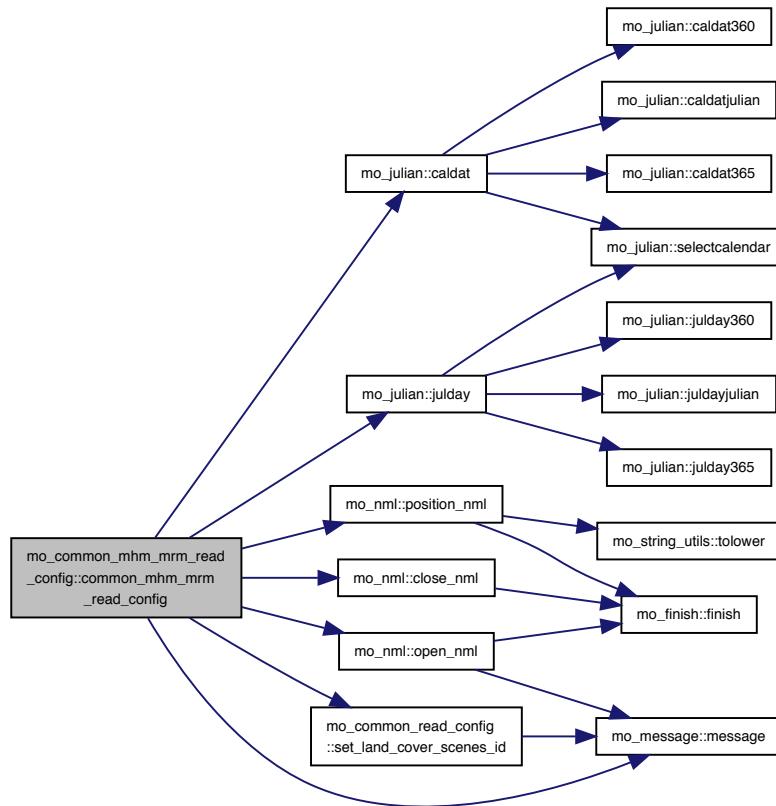
Date

Dec 2012

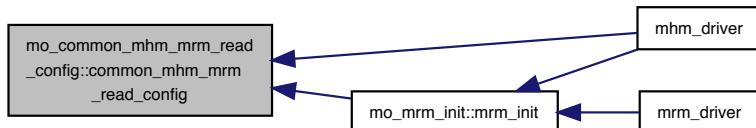
References mo_julian::caldat(), mo_nml::close_nml(), mo_common_mhm_mrm_variables::dds_r, mo_common_mhm_mrm_variables::dirrestartin, mo_common_mhm_mrm_variables::evalper, mo_julian::julday(), mo_common_variables::lcfilename, mo_common_mhm_mrm_variables::lcyearid, mo_common_constants::maxnobasins, mo_common_mhm_mrm_variables::mcmc_error_params, mo_common_mhm_mrm_variables::mcmc_opti, mo_message::message(), mo_common_variables::nbasins, mo_common_mhm_mrm_variables::niterations, mo_common_mhm_mrm_variables::ntstepday, mo_nml::open_nml(), mo_common_mhm_mrm_variables::opti_function, mo_common_mhm_mrm_variables::opti_method, mo_common_mhm_mrm_variables::optimize, mo_common_mhm_mrm_variables::optimize_restart, mo_nml::position_nml(), mo_common_mhm_mrm_variables::read_restart, mo_common_mhm_mrm_variables::resolutionrouting, mo_common_mhm_mrm_variables::sa_temp, mo_common_mhm_mrm_variables::sce_ngs, mo_common_mhm_mrm_variables::sce_npg, mo_common_mhm_mrm_variables::sce_nps, mo_common_mhm_mrm_variables::seed, mo_common_read_config::set_land_cover_scenes_id(), mo_common_mhm_mrm_variables::simper, mo_common_mhm_mrm_variables::timestep, mo_common_mhm_mrm_variables::warmingdays, and mo_common_mhm_mrm_variables::warmingper.

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:



15.11 mo_common_mhm_mrm_variables Module Reference

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

Variables

- integer(i4) [mrm_coupling_mode](#)
- integer(i4), public [timestep](#)

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

15.11.1 Detailed Description

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

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

Authors

Stephan Thober

Date

Sep 2015

15.11.2 Variable Documentation

15.11.2.1 dds_r

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

Referenced by `mo_common_mhm_mrm_read_config::check_optimization_settings()`, `mo_common_mhm_mrm::read_config::common_mhm_mrm_read_config()`, and `mo_optimization::optimization()`.

15.11.2.2 dirrestartin

```
character(256), dimension(:), allocatable, public mo_common_mhm_mrm_variables::dirrestartin
Referenced by mo_common_mhm_mrm_read_config::common_mhm_mrm_read_config(), mo_mhm_eval::mhm←
_eval(), mo_startup::mhm_initialize(), mo_mrm_eval::mrm_eval(), and mo_mrm_init::mrm_init().
```

15.11.2.3 evalper

```
type(period), dimension(:), allocatable, public mo_common_mhm_mrm_variables::evalper
Referenced by mo_common_mhm_mrm_read_config::common_mhm_mrm_read_config(), mo_mrm_write←
_fluxes_states::createoutputfile(), mo_write_fluxes_states::createoutputfile(), mo_objective_function::extract←
_basin_avg_tws(), mo_mrm_objective_function_runoff::extract_runoff(), mo_mrm_read_data::mrm_read←
discharge(), mo_mrm_write::mrm_write(), mo_mrm_objective_function_runoff::multi_objective_ae_fdc_lsv_nse←
_djf(), mo_objective_function::objective_kge_q_rmse_et(), mo_objective_function::objective_kge_q_rmse_tws(),
mo_read_optional_data::read_basin_avg_tws(), mo_read_optional_data::read_evapotranspiration(),
mo_read_optional_data::read_neutrons(), mo_read_optional_data::read_soil_moisture(),
mo_write_ascii::write_configfile(), mo_mrm_write::write_configfile(), and mo_mrm_write::write_daily_obs_sim_discharge().
```

15.11.2.4 lcyearid

```
integer(i4), dimension(:, :), allocatable, public mo_common_mhm_mrm_variables::lcyearid
Referenced by mo_common_mhm_mrm_read_config::common_mhm_mrm_read_config(), mo_mhm_eval::mhm←
_eval(), mo_mrm_eval::mrm_eval(), mo_write_ascii::write_configfile(), and mo_mrm_write::write_configfile().
```

15.11.2.5 mcmc_error_params

```
real(dp), dimension(nerror_model), public mo_common_mhm_mrm_variables::mcmc_error_params
Referenced by mo_common_mhm_mrm_read_config::common_mhm_mrm_read_config(), and mo_optimization::optimization().
```

15.11.2.6 mcmc_opti

```
logical, public mo_common_mhm_mrm_variables::mcmc_opti
Referenced by mo_common_mhm_mrm_read_config::common_mhm_mrm_read_config(), and mo_optimization::optimization().
```

15.11.2.7 mrm_coupling_mode

```
integer(i4) mo_common_mhm_mrm_variables::mrm_coupling_mode
Referenced by mrm_driver(), mo_mrm_init::mrm_init(), mo_mrm_write::mrm_write(), and mo_mrm_write::write←
configfile().
```

15.11.2.8 nerror_model

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

15.11.2.9 niterations

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

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

15.11.2.10 ntstepday

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

Referenced by mo_meteo_forcings::chunk_size(), mo_common_mhm_mrm_read_config::common_mhm_mrm_read_config(), mo_objective_function::extract_basin_avg_tws(), mo_mrm_objective_function_runoff::extract_runoff(), mo_write_fluxes_states::fluxesunit(), mo_meteo_forcings::is_read(), mhm_driver(), mo_mhm_eval::mhm_eval(), mo_mrm_eval::mrm_eval(), mo_mrm_read_data::mrm_read_discharge(), mo_mrm_write::mrm_write(), and mo_read_optional_data::read_basin_avg_tws().

15.11.2.11 opti_function

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

Referenced by mo_common_mhm_mrm_read_config::common_mhm_mrm_read_config(), mo_mhm_read_config::mhm_read_config(), mo_mrm_read_data::mrm_read_discharge(), mo_mrm_objective_function_runoff::multi_objective_runoff(), mo_objective_function::objective(), mo_optimization::optimization(), mo_read_optional_data::read_basin_avg_tws(), and mo_mrm_objective_function_runoff::single_objective_runoff().

15.11.2.12 opti_method

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

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

15.11.2.13 optimize

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

Referenced by mo_common_mhm_mrm_read_config::common_mhm_mrm_read_config(), mo_mhm_eval::mhm_eval(), mo_mhm_read_config::mhm_read_config(), mrm_driver(), mo_mrm_eval::mrm_eval(), mo_mrm_read_data::mrm_read_discharge(), and mo_read_optional_data::read_basin_avg_tws().

15.11.2.14 optimize_restart

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

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

15.11.2.15 `read_restart`

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

Referenced by `mo_common_mhm_mrm_read_config::common_mhm_mrm_read_config()`, `mo_mhm_eval::mhm_eval()`, `mo_startup::mhm_initialize()`, `mo_mrm_eval::mrm_eval()`, `mo_mrm_init::mrm_init()`, `mo_write_ascii::write_configfile()`, and `mo_mrm_write::write_configfile()`.

15.11.2.16 `readper`

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

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

15.11.2.17 `resolutionrouting`

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

Referenced by `mo_common_mhm_mrm_read_config::common_check_resolution()`, `mo_common_mhm_mrm_read_config::common_mhm_mrm_read_config()`, `mo_mhm_eval::mhm_eval()`, `mo_mrm_eval::mrm_eval()`, `mo_mrm_init::mrm_init()`, `mo_mrm_init::mrm_init_param()`, `mo_mrm_init::mrm_update_param()`, `mo_write_ascii::write_configfile()`, and `mo_mrm_write::write_configfile()`.

15.11.2.18 `sa_temp`

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

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

15.11.2.19 `sce_ngs`

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

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

15.11.2.20 `sce_npg`

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

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

15.11.2.21 sce_nps

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

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

15.11.2.22 seed

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

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

15.11.2.23 simper

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

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

15.11.2.24 timestep

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

Referenced by mo_common_mhm_mrm_read_config::common_mhm_mrm_read_config(), mo_startup::constants_init(), mo_write_fluxes_states::fluxesunit(), mo_meteo_forcings::is_read(), mo_mhm_eval::mhm_eval(), mo_mrm_eval::mrm_eval(), mo_mrm_init::mrm_init_param(), mo_mrm_read_data::mrm_read_total::runoff(), mo_write_ascii::write_configfile(), and mo_mrm_write::write_configfile().

15.11.2.25 warmingdays

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

Referenced by mo_common_mhm_mrm_read_config::common_mhm_mrm_read_config(), mo_objective::function::extract_basin_avg_tws(), mo_mrm_objective_function::runoff::extract_runoff(), mo_mhm_eval::mhm::eval(), mo_mrm_eval::mrm_eval(), and mo_mrm_write::mrm_write().

15.11.2.26 warmper

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

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

15.12 mo_common_read_config Module Reference

Reading of main model configurations.

Functions/Subroutines

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

15.12.1 Detailed Description

Reading of main model configurations.

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

Authors

Matthias Zink

Date

Dec 2012

15.12.2 Function/Subroutine Documentation

15.12.2.1 common_read_config()

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

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

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

Parameters

in	<code>character(*) :: file_namelist</code>	name of file
in	<code>integer :: unamelist</code>	id of file

Authors

Matthias Zink

Date

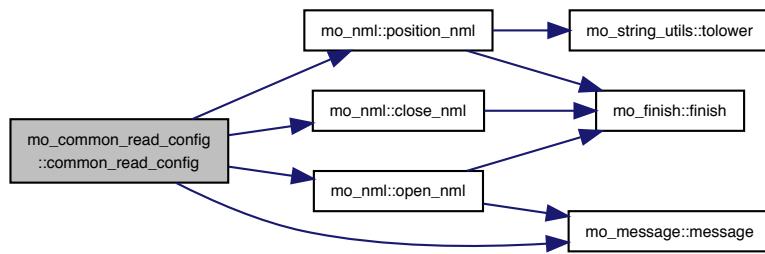
Dec 2012

References `mo_nml::close_nml()`, `mo_common_variables::contact`, `mo_common_variables::conventions`, `mo_common_variables::dircommonfiles`, `mo_common_variables::dirconfigout`, `mo_common_variables::dirlcov`,

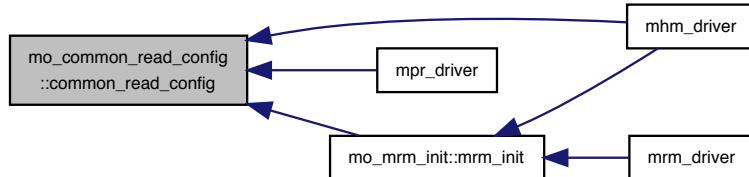
mo_common_variables::dirmorpho, mo_common_variables::dirout, mo_common_variables::dirrestartout, mo_common_variables::filelatlon, mo_common_variables::history, mo_common_variables::iflag_coordinate_sys, mo_common_variables::l0_basin, mo_common_variables::lc_year_end, mo_common_variables::lc_year_start, mo_common_variables::lcfilename, mo_common_constants::maxnlcovers, mo_common_constants::maxnobasins, mo_message::message(), mo_common_variables::mhmm_details, mo_common_variables::nbasins, mo_common_variables::nlcoverscene, mo_common_variables::nprocesses, mo_common_variables::nunique10basins, mo_nml::open_nml(), mo_nml::position_nml(), mo_common_variables::processmatrix, mo_common_variables::project_details, mo_common_variables::resolutionhydrology, mo_common_variables::setup_description, mo_common_variables::simulation_type, and mo_common_variables::write_restart.

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

Here is the call graph for this function:



Here is the caller graph for this function:



15.12.2.2 set_land_cover_scenes_id()

```

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

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

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

Parameters

in	<i>type(period), dimension(:) :: sim_Per</i>	
in, out	<i>integer(i4), dimension(:, :) :: LCyear_Id</i>	
in, out	<i>character(256), dimension(:) :: LCfilename</i>	

Authors

Matthias Zink

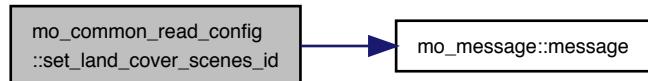
Date

Dec 2012

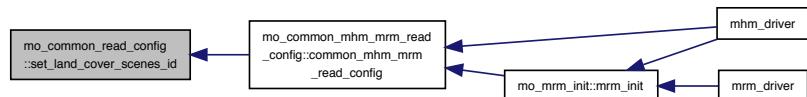
References `mo_common_variables::lc_year_end`, `mo_common_variables::lc_year_start`, `mo_message::message()`, `mo_common_variables::nbasins`, `mo_common_variables::nlcoverscene`, and `mo_common_constants::nodata_i4`.

Referenced by `mo_common_mhm_mrm_read_config::common_mhm_mrm_read_config()`.

Here is the call graph for this function:



Here is the caller graph for this function:



15.13 mo_common_read_data Module Reference

TODO: add description.

Functions/Subroutines

- subroutine, public `read_dem`
`TODO: add description.`
- subroutine, public `read_lcov`
`TODO: add description.`

15.13.1 Detailed Description

TODO: add description.

TODO: add description

Authors

Robert Schwepppe

Date

Jun 2018

15.13.2 Function/Subroutine Documentation

15.13.2.1 read_dem()

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

TODO: add description.

TODO: add description

Authors

Robert Schwepppe

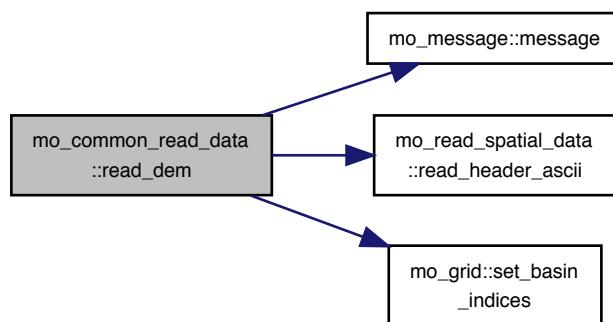
Date

Jun 2018

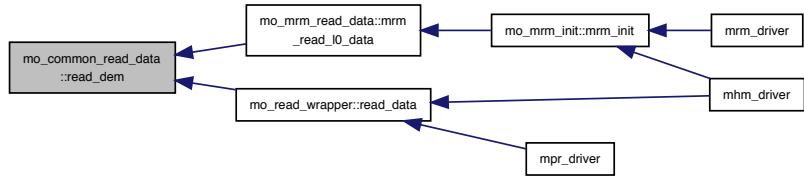
References mo_common_variables::dirmorpho, mo_common_file::file_dem, mo_common_variables::l0_basin, mo_common_variables::l0_elev, mo_common_variables::level0, mo_message::message(), mo_common_variables::nbasins, mo_common_constants::nodata_dp, mo_common_variables::nuniqueL0basins, mo_read_spatial_data::read_header_ascii(), mo_common_variables::resolutionhydrology, mo_grid::set_basin_indices(), and mo_common_file::udem.

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

Here is the call graph for this function:



Here is the caller graph for this function:



15.13.2.2 read_lcover()

subroutine, public mo_common_read_data::read_lcover ()

TODO: add description.

TODO: add description

Authors

Robert Scheweppe

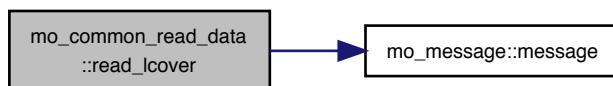
Date

Jun 2018

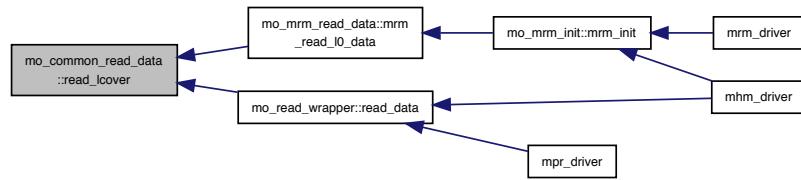
References mo_common_variables::dirlcover, mo_common_variables::l0_basin, mo_common_variables::l0_lcover, mo_common_variables::lcfilename, mo_common_variables::level0, mo_message::message(), mo_common_variables::nbasins, mo_common_variables::nlcoverscene, mo_common_constants::nodata_i4, and mo_common_file::ulcoverclass.

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

Here is the call graph for this function:



Here is the caller graph for this function:



15.14 mo_common_restart Module Reference

TODO: add description.

Functions/Subroutines

- subroutine, public [write_grid_info](#) (grid_in, level_name, nc)
write restart files for each basin
- subroutine, public [read_grid_info](#) (iBasin, InPath, level_name, fname_part, new_grid)
reads configuration apart from Level 11 configuration from a restart directory

15.14.1 Detailed Description

TODO: add description.

TODO: add description

Authors

Robert Schwepppe

Date

Jun 2018

15.14.2 Function/Subroutine Documentation

15.14.2.1 [read_grid_info\(\)](#)

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

reads configuration apart from Level 11 configuration from a restart directory

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

Parameters

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

Authors

Stephan Thober

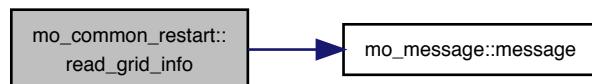
Date

Apr 2013

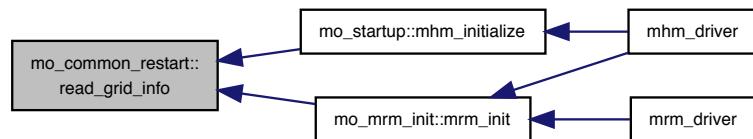
References `mo_kind::dp`, `mo_kind::i4`, and `mo_message::message()`.

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

Here is the call graph for this function:



Here is the caller graph for this function:

15.14.2.2 `write_grid_info()`

```

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

write restart files for each basin

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

Parameters

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

Authors

Stephan Thober

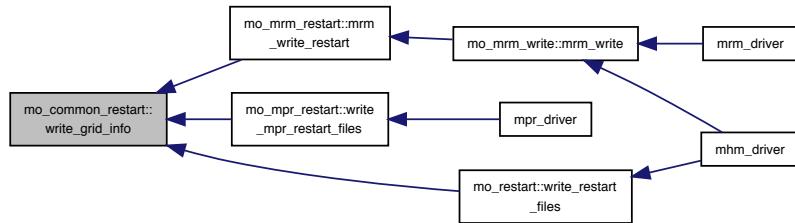
Date

Jun 2014

References `mo_kind::dp`, `mo_kind::i4`, `mo_common_constants::nodata_dp`, and `mo_common_constants::nodata←_i4`.

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

Here is the caller graph for this function:



15.15 mo_common_variables Module Reference

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

Data Types

- type `grid`
- type `gridremapper`
- type `period`

Variables

- character(1024), public `project_details`

- character(1024), public `setup_description`
- character(1024), public `simulation_type`
- character(256), public `conventions`
- character(1024), public `contact`
- character(1024), public `mhm_details`
- character(1024), public `history`
- integer(i4), public `iflag_cordinate_sys`
- real(dp), dimension(:), allocatable, public `resolutionhydrology`
- integer(i4), dimension(:), allocatable, public `l0_basin`
- 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 `dircover`
- character(256), dimension(:), allocatable, public `dirout`
- character(256), dimension(:), allocatable, public `filelatlon`
- type(`grid`), dimension(:), allocatable, target, public `level0`
- type(`grid`), dimension(:), allocatable, target, public `level1`
- type(`gridremapper`), dimension(:), allocatable, public `l0_l1_remap`
- real(dp), dimension(:), allocatable, public `l0_elev`
- integer(i4), dimension(:, :), allocatable, public `l0_lcover`
- 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`

15.15.1 Detailed Description

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

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

Authors

Stephan Thober

Date

Sep 2015

15.15.2 Variable Documentation

15.15.2.1 alma_convention

```
logical mo_common_variables::alma_convention
```

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

15.15.2.2 contact

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

Referenced by mo_common_read_config::common_read_config().

15.15.2.3 conventions

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

Referenced by mo_common_read_config::common_read_config().

15.15.2.4 dircommonfiles

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

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

15.15.2.5 dirconfigout

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

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

15.15.2.6 dirlcover

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

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

15.15.2.7 dirmorpho

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

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

15.15.2.8 dirout

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

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

15.15.2.9 dirrestartout

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

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

15.15.2.10 filelatlon

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

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

15.15.2.11 global_parameters

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

Referenced by mo_common_mhm_mrm_read_config::check_optimization_settings(), mo_mrm_objective_function_runoff::loglikelihood_evin2013_2(), mo_multi_param_reg::mpr(), mo_mpr_read_config::mpr_read_config(), mrm_driver(), mo_mrm_init::mrm_init(), mo_optimization::optimization(), mo_read_wrapper::read_data(), mo_mrm_read_config::read_mrm_routing_params(), mo_write_ascii::write_configfile(), and mo_mrm_write::write_configfile().

15.15.2.12 global_parameters_name

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

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

15.15.2.13 history

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

Referenced by mo_common_read_config::common_read_config().

15.15.2.14 iflag_cordinate_sys

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

Referenced by `mo_common_read_config::common_read_config()`, `mo_grid::l0_grid_setup()`, `mo_mrm_net::startup::l11_stream_features()`, `mo_mrm_init::mrm_init_param()`, `mo_mrm_init::mrm_update_param()`, and `mo_write_ascii::write_configfile()`.

15.15.2.15 l0_basin

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

Referenced by `mo_mrm_net_startup::celllength()`, `mo_common_read_config::common_read_config()`, `mo_mrm_net_startup::l11_flow_direction()`, `mo_mrm_net_startup::l11_fraction_sealed_floodplain()`, `mo_mrm_net_startup::l11_link_location()`, `mo_mrm_net_startup::l11_set_drain_outlet_gauges()`, `mo_mrm_net_startup::l11_stream_features()`, `mo_startup::mhmm_initialize()`, `mo_mpr_eval::mpr_eval()`, `mo_mpr_startup::mpr_initialize()`, `mo_mrm_init::mrm_init()`, `mo_mrm_read_data::mrm_read_l0_data()`, `mo_read_wrapper::read_data()`, `mo_common_read_data::read_dem()`, `mo_common_read_data::read_lcover()`, `mo_write_ascii::write_configfile()`, and `mo_mrm_write::write_configfile()`.

15.15.2.16 l0_elev

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

Referenced by `mo_mpr_startup::l0_check_input()`, `mo_mrm_net_startup::l11_stream_features()`, and `mo_common_read_data::read_dem()`.

15.15.2.17 l0_l1_remap

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

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

15.15.2.18 l0_lcover

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

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

15.15.2.19 lc_year_end

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

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

15.15.2.20 lc_year_start

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

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

15.15.2.21 lcfilename

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

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

15.15.2.22 level0

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

Referenced by `mo_mrm_net_startup::celllength()`, `mo_mpr_startup::l0_check_input()`, `mo_mrm_init::l0_check_input_routing()`, `mo_mpr_startup::l0_variable_init()`, `mo_mrm_net_startup::l11_flow_direction()`, `mo_mrm_net_startup::l11_fraction_sealed_floodplain()`, `mo_mrm_net_startup::l11_link_location()`, `mo_mrm_net_startup::l11_set_drain_outlet_gauges()`, `mo_mrm_net_startup::l11_stream_features()`, `mo_startup::mhm_initialize()`, `mo_mpr_eval::mpr_eval()`, `mo_mpr_startup::mpr_initialize()`, `mo_mrm_init::mrm_init()`, `mo_mrm_read_data::mrm_read_l0_data()`, `mo_read_wrapper::read_data()`, `mo_common_read_data::read_dem()`, `mo_common_read_data::read_lccover()`, `mo_write_ascii::write_configfile()`, and `mo_mrm_write::write_configfile()`.

15.15.2.23 level1

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

Referenced by `mo_write_fluxes_states::createoutputfile()`, `mo_mrm_net_startup::l11_l1_mapping()`, `mo_meteo_forcings::meteo_forcings_wrapper()`, `mo_meteo_forcings::meteo_weights_wrapper()`, `mo_mhm_eval::mhm_eval()`, `mo_startup::mhm_initialize()`, `mo_mpr_eval::mpr_eval()`, `mo_mpr_startup::mpr_initialize()`, `mo_mrm_eval::mrm_eval()`, `mo_mrm_init::mrm_init()`, `mo_mrm_restart::mrm_read_restart_config()`, `mo_mrm_read_data::mrm_read_total_runoff()`, `mo_mrm_restart::mrm_write_restart()`, `mo_objective_function::objective_et_kge_catchment_avg()`, `mo_objective_function::objective_kge_q_et()`, `mo_objective_function::objective_kge_q_rmse_et()`, `mo_objective_function::objective_kge_q_sm_corr()`, `mo_objective_function::objective_neutrons_kge_catchment_avg()`, `mo_objective_function::objective_sm_corr()`, `mo_objective_function::objective_sm_kge_catchment_avg()`, `mo_objective_function::objective_sm_pd()`, `mo_objective_function::objective_sm_sse_standard_score()`, `mo_read_optional_data::read_evapotranspiration()`, `mo_read_optional_data::read_neutrons()`, `mo_restart::read_restart_states()`, `mo_read_optional_data::read_soil_moisture()`, `mo_write_ascii::write_configfile()`, `mo_mrm_write::write_configfile()`, `mo_mpr_restart::write_mpr_restart_files()`, and `mo_restart::write_restart_files()`.

15.15.2.24 mhm_details

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

Referenced by `mo_common_read_config::common_read_config()`.

15.15.2.25 nbasins

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

Referenced by mo_common_mhm_mrm_read_config::common_check_resolution(), mo_common_mhm_mrm::read_config::common_mhm_mrm_read_config(), mo_common_read_config::common_read_config(), mo::mrm_init::config_output(), mo_mrm_net_startup::l11_fraction_sealed_floodplain(), mo_mhm_eval::mhm_eval(), mo_startup::mhm_initialize(), mo_mhm_read_config::mhm_read_config(), mo_mpr_eval::mpr_eval(), mo_mpr::startup::mpr_initialize(), mo_mpr_read_config::mpr_read_config(), mo_mrm_eval::mrm_eval(), mo_mrm_init::mrm_init(), mo_mrm_init::mrm_init_param(), mo_mrm_read_config::mrm_read_config(), mo_mrm_read_data::mrm_read_discharge(), mo_mrm_read_data::mrm_read_l0_data(), mo_mrm_restart::mrm_read_restart_config(), mo_mrm_write::mrm_write(), mo_objective_function::objective_et_kge_catchment_avg(), mo_objective_function::objective_kge_q_et(), mo_objective_function::objective_kge_q_rmse_et(), mo_objective_function::objective_kge_q_rmse_tws(), mo_objective_function::objective_kge_q_sm_corr(), mo_objective_function::objective_neutrons_kge_catchment_avg(), mo_objective_function::objective_sm_corr(), mo_objective_function::objective_sm_kge_catchment_avg(), mo_objective_function::objective_sm_pd(), mo_objective_function::objective_sm_sse_standard_score(), mo_meteo_forcings::prepare_meteo_forcings_data(), mo_read_optional_data::read_basin_avg_tws(), mo_read_wrapper::read_data(), mo_common_read_data::read_dem(), mo_common_read_data::read_lcover(), mo_common_read_config::set_land_cover_scenes_id(), mo_write_ascii::write_configfile(), mo_mrm_write::write_configfile(), and mo_mrm_write::write_daily_obs_sim_discharge().

15.15.2.26 nlcoverscene

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

Referenced by mo_common_read_config::common_read_config(), mo_mpr_startup::init_eff_params(), mo_mpr::startup::l0_check_input(), mo_mrm_net_startup::l11_fraction_sealed_floodplain(), mo_mrm_restart::mrm_read::restart_states(), mo_mrm_restart::mrm_write_restart(), mo_common_read_data::read_lcover(), mo_restart::read_restart_states(), mo_common_read_config::set_land_cover_scenes_id(), mo_write_ascii::write_configfile(), mo::mrm_write::write_configfile(), mo_mpr_restart::write_mpr_restart_files(), and mo_restart::write_restart_files().

15.15.2.27 nprocesses

```
integer(i4), parameter, public mo_common_variables::nprocesses = 10
```

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

15.15.2.28 nunique10basins

```
integer(i4), public mo_common_variables::nunique10basins
```

Referenced by mo_common_read_config::common_read_config(), and mo_common_read_data::read_dem().

15.15.2.29 processmatrix

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

Referenced by mo_common_read_config::common_read_config(), mo_startup::constants_init(), mo_mrm_net::startup::l11_stream_features(), mo_mhm_eval::mhm_eval(), mo_mhm_read_config::mhm_read_config(), mo::multi_param_reg::mpr(), mo_mpr_read_config::mpr_read_config(), mo_mrm_eval::mrm_eval(), mo_mrm_init::mrm_init(), mo_mrm_init::mrm_init_param(), mo_mrm_read_config::mrm_read_config(), mo_mrm_read_data::mrm_read_l0_data(), mo_mrm_restart::mrm_read_restart_config(), mo_mrm_write::mrm_write_optinamelist(),

mo_mrm_restart::mrm_write_restart(), mo_meteo_forcings::prepare_meteo_forcings_data(), mo_read_wrapper::read_data(), mo_mrm_read_config::read_mrm_routing_params(), mo_restart::read_restart_states(), mo_write_ascii::write_configfile(), mo_mrm_write::write_configfile(), and mo_mpr_restart::write_eff_params().

15.15.2.30 project_details

character(1024), public mo_common_variables::project_details

Referenced by mo_common_read_config::common_read_config().

15.15.2.31 resolutionhydrology

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

Referenced by mo_common_mhm_mrm_read_config::common_check_resolution(), mo_common_read_config::common_read_config(), mo_mhm_eval::mhm_eval(), mo_mpr_startup::mpr_initialize(), mo_mrm_eval::mrm_eval(), mo_mrm_init::mrm_init(), mo_common_read_data::read_dem(), mo_write_ascii::write_configfile(), and mo_mrm_write::write_configfile().

15.15.2.32 setup_description

character(1024), public mo_common_variables::setup_description

Referenced by mo_common_read_config::common_read_config().

15.15.2.33 simulation_type

character(1024), public mo_common_variables::simulation_type

Referenced by mo_common_read_config::common_read_config().

15.15.2.34 write_restart

logical, public mo_common_variables::write_restart

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

15.16 mo_constants Module Reference

Provides computational, mathematical, physical, and file constants.

Variables

- real(dp), parameter **pi_dp** = 3.141592653589793238462643383279502884197_dp
Pi in double precision.
- real(sp), parameter **pi_sp** = 3.141592653589793238462643383279502884197_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 atmosphere 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` = 3.141592653589793238462643383279502884197_dp

Pi in double precision.

- `real(sp)`, parameter `pi` = 3.141592653589793238462643383279502884197_sp

Pi in single precision.

- `real(dp)`, parameter `pio2_d` = 1.57079632679489661923132169163975144209858_dp

Pi/2 in double precision.

- `real(sp)`, parameter `pio2` = 1.57079632679489661923132169163975144209858_sp

Pi/2 in single precision.

- `real(dp)`, parameter `twopi_d` = 6.283185307179586476925286766559005768394_dp

*2*Pi in double precision*

- `real(sp)`, parameter `twopi` = 6.283185307179586476925286766559005768394_sp

*2*Pi in single precision*

- `real(dp)`, parameter `sqrt2_d` = 1.41421356237309504880168872420969807856967_dp

Square root of 2 in double precision.

- `real(sp)`, parameter `sqrt2` = 1.41421356237309504880168872420969807856967_sp

Square root of 2 in single precision.

- `real(dp)`, parameter `euler_d` = 0.5772156649015328606065120900824024310422_dp

Euler's constant in double precision.

- `real(sp)`, parameter `euler` = 0.5772156649015328606065120900824024310422_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.

15.16.1 Detailed Description

Provides computational, mathematical, physical, and file constants.

Provides computational constants like epsilon, mathematical constants such as Pi, physical constants such as the Stefan-Boltzmann constant, and file units for some standard streams such as standard in.

Author

Matthias Cuntz

Date

Nov 2011

15.16.2 Variable Documentation

15.16.2.1 cp0_dp

real(dp), parameter mo_constants::cp0_dp = 1005.0_dp

specific heat capacity of air [J kg⁻¹ K⁻¹] in double precision

Referenced by mo_pet::pet_penman().

15.16.2.2 cp0_sp

real(sp), parameter mo_constants::cp0_sp = 1005.0_sp

specific heat capacity of air [J kg⁻¹ K⁻¹] in single precision

15.16.2.3 dayhours

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

Referenced by mo_read_forcing_nc::get_time_vector_and_select().

15.16.2.4 daysecs

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

Referenced by mo_read_forcing_nc::get_time_vector_and_select().

15.16.2.5 deg2rad_dp

real(dp), parameter mo_constants::deg2rad_dp = PI_dp / 180._dp

degree to radian conversion (pi/180) in double precision

Referenced by mo_pet::pet_hargreaves().

15.16.2.6 deg2rad_sp

```
real(sp), parameter mo_constants::deg2rad_sp = PI_sp / 180._sp
degree to radian conversion (pi/180) in double precision
```

15.16.2.7 euler

```
real(sp), parameter mo_constants::euler = 0.5772156649015328606065120900824024310422_sp
Euler's constant in single precision.
```

15.16.2.8 euler_d

```
real(dp), parameter mo_constants::euler_d = 0.5772156649015328606065120900824024310422_dp
Euler's constant in double precision.
```

15.16.2.9 gravity_dp

```
real(dp), parameter mo_constants::gravity_dp = 9.81_dp
Gravity acceleration [m^2 s^-1] in double precision.
```

15.16.2.10 gravity_sp

```
real(sp), parameter mo_constants::gravity_sp = 9.81_sp
Gravity acceleration [m^2 s^-1] in single precision.
```

15.16.2.11 nerr

```
integer, parameter mo_constants::nerr = error_unit
Standard error file unit.
```

15.16.2.12 nin

```
integer, parameter mo_constants::nin = input_unit
Standard input file unit.
```

15.16.2.13 nnml

```
integer, parameter mo_constants::nnml = 100
Standard file unit for namelist.
```

15.16.2.14 nout

```
integer, parameter mo_constants::nout = output_unit
```

Standard output file unit.

Referenced by mo_message::message().

15.16.2.15 p0_dp

```
real(dp), parameter mo_constants::p0_dp = 101325._dp
```

standard atmosphere Standard pressure [Pa] in double precision

15.16.2.16 p0_sp

```
real(sp), parameter mo_constants::p0_sp = 101325._sp
```

Standard pressure [Pa] in single precision.

15.16.2.17 pi

```
real(sp), parameter mo_constants::pi = 3.141592653589793238462643383279502884197_sp
```

Pi in single precision.

15.16.2.18 pi_d

```
real(dp), parameter mo_constants::pi_d = 3.141592653589793238462643383279502884197_dp
```

Pi in double precision.

Referenced by mo_pet::extraterr_rad_approx().

15.16.2.19 pi_dp

```
real(dp), parameter mo_constants::pi_dp = 3.141592653589793238462643383279502884197_dp
```

Pi in double precision.

Referenced by mo_neutrons::cosmic(), mo_corr::fourrow_dp(), mo_corr::fourrow_sp(), mo_mrm_objective_function_runoff::loglikelihood_evin2013_2(), mo_neutrons::lookupintegral(), mo_neutrons::oldintegration(), mo_mrm_objective_function_runoff::parameter_regularization(), and mo_neutrons::tabularintegralafast().

15.16.2.20 pi_sp

```
real(sp), parameter mo_constants::pi_sp = 3.141592653589793238462643383279502884197_sp
```

Pi in single precision.

15.16.2.21 pio2

```
real(sp), parameter mo_constants::pio2 = 1.57079632679489661923132169163975144209858_sp
```

Pi/2 in single precision.

15.16.2.22 pio2_d

```
real(dp), parameter mo_constants::pio2_d = 1.57079632679489661923132169163975144209858_dp
```

Pi/2 in double precision.

15.16.2.23 pio2_dp

```
real(dp), parameter mo_constants::pio2_dp = 1.57079632679489661923132169163975144209858_dp
```

Pi/2 in double precision.

15.16.2.24 pio2_sp

```
real(sp), parameter mo_constants::pio2_sp = 1.57079632679489661923132169163975144209858_sp
```

Pi/2 in single precision.

15.16.2.25 psychro_dp

```
real(dp), parameter mo_constants::psychro_dp = 0.0646_dp
```

Psychrometric constant [kPa K⁻¹] in double precision.

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

15.16.2.26 psychro_sp

```
real(sp), parameter mo_constants::psychro_sp = 0.0646_sp
```

Psychrometric constant [kPa K⁻¹] in single precision.

15.16.2.27 rad2deg_dp

```
real(dp), parameter mo_constants::rad2deg_dp = 180._dp / PI_dp
```

radian to conversion (180/pi) in double precision

15.16.2.28 rad2deg_sp

```
real(sp), parameter mo_constants::rad2deg_sp = 180._sp / PI_sp
radian to degree conversion (180/pi) in single precision
```

15.16.2.29 radiusearth_dp

```
real(dp), parameter mo_constants::radiusearth_dp = 6371228._dp
```

Referenced by mo_mrm_net_startup::get_distance_two_lat_lon_points(), and mo_grid::l0_grid_setup().

15.16.2.30 radiusearth_sp

```
real(sp), parameter mo_constants::radiusearth_sp = 6371228._sp
```

15.16.2.31 rho0_dp

```
real(dp), parameter mo_constants::rho0_dp = 1.225_dp
```

standard density [kg m^-3] in double precision

Referenced by mo_pet::pet_penman().

15.16.2.32 rho0_sp

```
real(sp), parameter mo_constants::rho0_sp = 1.225_sp
```

standard density [kg m^-3] in single precision

15.16.2.33 secday_dp

```
real(dp), parameter, public mo_constants::secday_dp = 86400.0_dp
```

15.16.2.34 secday_sp

```
real(sp), parameter, public mo_constants::secday_sp = 86400.0_sp
```

Time conversion Seconds per day [s] in single precision.

15.16.2.35 sigma_dp

```
real(dp), parameter mo_constants::sigma_dp = 5.67e-08_dp
```

Stefan-Boltzmann constant [W m^-2 K^-4] in double precision.

15.16.2.36 sigma_sp

```
real(sp), parameter mo_constants::sigma_sp = 5.67e-08_sp
```

Stefan-Boltzmann constant [W m⁻² K⁻⁴] in single precision.

15.16.2.37 solarconst_dp

```
real(dp), parameter mo_constants::solarconst_dp = 1367._dp
```

Solar constant in [J m⁻² s⁻¹] in double precision.

Referenced by mo_pet::extraterr_rad_approx().

15.16.2.38 solarconst_sp

```
real(sp), parameter mo_constants::solarconst_sp = 1367._sp
```

Solar constant in [J m⁻² s⁻¹] in single precision.

15.16.2.39 specheatet_dp

```
real(dp), parameter mo_constants::specheatet_dp = 2.45e06_dp
```

Specific heat for vaporization of water in [J m⁻² mm⁻¹] in double precision.

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

15.16.2.40 specheatet_sp

```
real(sp), parameter mo_constants::specheatet_sp = 2.45e06_sp
```

Specific heat for vaporization of water in [J m⁻² mm⁻¹] in single precision.

15.16.2.41 sqrt2

```
real(sp), parameter mo_constants::sqrt2 = 1.41421356237309504880168872420969807856967_sp
```

Square root of 2 in single precision.

15.16.2.42 sqrt2_d

```
real(dp), parameter mo_constants::sqrt2_d = 1.41421356237309504880168872420969807856967_dp
```

Square root of 2 in double precision.

15.16.2.43 sqrt2_dp

```
real(dp), parameter mo_constants::sqrt2_dp = 1.41421356237309504880168872420969807856967_dp
```

Square root of 2 in double precision.

Referenced by mo_mrm_net_startup::celllength().

15.16.2.44 sqrt2_sp

```
real(sp), parameter mo_constants::sqrt2_sp = 1.41421356237309504880168872420969807856967_sp
```

Square root of 2 in single precision.

15.16.2.45 t0_dp

```
real(dp), parameter mo_constants::t0_dp = 273.15_dp
```

Standard temperature [K] in double precision.

15.16.2.46 t0_sp

```
real(sp), parameter mo_constants::t0_sp = 273.15_sp
```

Standard temperature [K] in single precision.

15.16.2.47 twopi

```
real(sp), parameter mo_constants::twopi = 6.283185307179586476925286766559005768394_sp
```

2*Pi in single precision

15.16.2.48 twopi_d

```
real(dp), parameter mo_constants::twopi_d = 6.283185307179586476925286766559005768394_dp
```

2*Pi in double precision

Referenced by mo_pet::extraterr_rad_approx().

15.16.2.49 twopi_dp

```
real(dp), parameter mo_constants::twopi_dp = 6.283185307179586476925286766559005768394_dp
```

2*Pi in double precision

Referenced by mo_corr::four1_dp(), mo_corr::four1_sp(), mo_mrm_net_startup::get_distance_two_lat_lon_←
points(), mo_grid::l0_grid_setup(), and mo_corr::zroots_unity_dp().

15.16.2.50 twopi_sp

```
real(sp), parameter mo_constants::twopi_sp = 6.283185307179586476925286766559005768394_sp
```

2*Pi in single precision

Referenced by mo_corr::zroots_unity_sp().

15.16.2.51 twothird_dp

```
real(dp), parameter mo_constants::twothird_dp = 0.666666666666666666666666666666666666667_dp
```

2/3 in double precision

Referenced by mo_canopy_interc::canopy_interc().

15.16.2.52 twothird_sp

```
real(sp), parameter mo_constants::twothird_sp = 0.66666666666666666666666666666666666667_sp
```

2/3 in single precision

15.16.2.53 yeardays

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

Referenced by mo_read_forcing_nc::get_time_vector_and_select().

15.16.2.54 yearmonths

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

15.17 mo_corr Module Reference

Data Types

- interface [arth](#)
- interface [autocoeffk](#)
- interface [autocorr](#)
- interface [corr](#)
- interface [crosscoeffk](#)
- interface [crosscorr](#)
- interface [four1](#)
- interface [fourrow](#)
- interface [realft](#)
- interface [swap](#)

Functions/Subroutines

- real(sp) function, dimension(n) `arth_sp` (first, increment, n)
- real(dp) function, dimension(n) `arth_dp` (first, increment, n)
- integer(i4) function, dimension(n) `arth_i4` (first, increment, n)
- real(dp) function `autocoeffk_dp` (x, k, mask)
- real(sp) function `autocoeffk_sp` (x, k, mask)
- real(dp) function, dimension(size(k)) `autocoeffk_1d_dp` (x, k, mask)
- real(sp) function, dimension(size(k)) `autocoeffk_1d_sp` (x, k, mask)
- real(dp) function `autocorr_dp` (x, k, mask)
- real(sp) function `autocorr_sp` (x, k, mask)
- real(dp) function, dimension(size(k)) `autocorr_1d_dp` (x, k, mask)
- real(sp) function, dimension(size(k)) `autocorr_1d_sp` (x, k, mask)
- real(dp) function, dimension(size(data1)) `corr_dp` (data1, data2, nadjust, nhigh, nwin)
- real(sp) function, dimension(size(data1)) `corr_sp` (data1, data2, nadjust, nhigh, nwin)
- real(dp) function `crosscoeffk_dp` (x, y, k, mask)
- real(sp) function `crosscoeffk_sp` (x, y, k, mask)
- real(dp) function `crosscorr_dp` (x, y, k, mask)
- real(sp) function `crosscorr_sp` (x, y, k, mask)
- subroutine `four1_sp` (data, isign)
- subroutine `four1_dp` (data, isign)
- subroutine `fourrow_sp` (data, isign)
- subroutine `fourrow_dp` (data, isign)
- subroutine `realft_dp` (data, isign, zdata)
- subroutine `realft_sp` (data, isign, zdata)
- subroutine `swap_1d_spc` (a, b)
- subroutine `swap_1d_dpc` (a, b)
- complex(dpc) function, dimension(nn) `zroots_unity_dp` (n, nn)
- complex(spc) function, dimension(nn) `zroots_unity_sp` (n, nn)

Variables

- integer(i4), parameter `npar_arth` = 16
- integer(i4), parameter `npar2_arth` = 8

15.17.1 Function/Subroutine Documentation

15.17.1.1 `arth_dp()`

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

References `npar2_arth`, and `npar_arth`.

15.17.1.2 arth_i4()

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

References npar2_arth, and npar_arth.

15.17.1.3 arth_sp()

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

References npar2_arth, and npar_arth.

15.17.1.4 autocoeffk_1d_dp()

```
real(dp) function, dimension(size(k)) mo_corr::autocoeffk_1d_dp (
    real(dp), dimension(:), intent(in) x,
    integer(i4), dimension(:), intent(in) k,
    logical, dimension(:), intent(in), optional mask ) [private]
```

15.17.1.5 autocoeffk_1d_sp()

```
real(sp) function, dimension(size(k)) mo_corr::autocoeffk_1d_sp (
    real(sp), dimension(:), intent(in) x,
    integer(i4), dimension(:), intent(in) k,
    logical, dimension(:), intent(in), optional mask ) [private]
```

15.17.1.6 autocoeffk_dp()

```
real(dp) function mo_corr::autocoeffk_dp (
    real(dp), dimension(:), intent(in) x,
    integer(i4), intent(in) k,
    logical, dimension(:), intent(in), optional mask ) [private]
```

15.17.1.7 autocoeffk_sp()

```
real(sp) function mo_corr::autocoeffk_sp (
    real(sp), dimension(:), intent(in) x,
    integer(i4), intent(in) k,
    logical, dimension(:), intent(in), optional mask ) [private]
```

15.17.1.8 autocorr_1d_dp()

```
real(dp) function, dimension(size(k)) mo_corr::autocorr_1d_dp (
    real(dp), dimension(:), intent(in) x,
    integer(i4), dimension(:), intent(in) k,
    logical, dimension(:), intent(in), optional mask ) [private]
```

15.17.1.9 autocorr_1d_sp()

```
real(sp) function, dimension(size(k)) mo_corr::autocorr_1d_sp (
    real(sp), dimension(:), intent(in) x,
    integer(i4), dimension(:), intent(in) k,
    logical, dimension(:), intent(in), optional mask ) [private]
```

15.17.1.10 autocorr_dp()

```
real(dp) function mo_corr::autocorr_dp (
    real(dp), dimension(:), intent(in) x,
    integer(i4), intent(in) k,
    logical, dimension(:), intent(in), optional mask ) [private]
```

15.17.1.11 autocorr_sp()

```
real(sp) function mo_corr::autocorr_sp (
    real(sp), dimension(:), intent(in) x,
    integer(i4), intent(in) k,
    logical, dimension(:), intent(in), optional mask ) [private]
```

15.17.1.12 corr_dp()

```
real(dp) function, dimension(size(data1)) mo_corr::corr_dp (
    real(dp), dimension(:), intent(in) data1,
    real(dp), dimension(:), intent(in) data2,
    integer(i4), intent(out), optional nadjust,
    integer(i4), intent(in), optional nhigh,
    integer(i4), intent(in), optional nwin ) [private]
```

15.17.1.13 corr_sp()

```
real(sp) function, dimension(size(data1)) mo_corr::corr_sp (
    real(sp), dimension(:), intent(in) data1,
    real(sp), dimension(:), intent(in) data2,
    integer(i4), intent(out), optional nadjust,
    integer(i4), intent(in), optional nhigh,
    integer(i4), intent(in), optional nwin ) [private]
```

15.17.1.14 crosscoeffk_dp()

```
real(dp) function mo_corr::crosscoeffk_dp (
    real(dp), dimension(:), intent(in) x,
    real(dp), dimension(:), intent(in) y,
    integer(i4), intent(in) k,
    logical, dimension(:), intent(in), optional mask ) [private]
```

15.17.1.15 crosscoeffk_sp()

```
real(sp) function mo_corr::crosscoeffk_sp (
    real(sp), dimension(:), intent(in) x,
    real(sp), dimension(:), intent(in) y,
    integer(i4), intent(in) k,
    logical, dimension(:), intent(in), optional mask ) [private]
```

15.17.1.16 crosscorr_dp()

```
real(dp) function mo_corr::crosscorr_dp (
    real(dp), dimension(:), intent(in) x,
    real(dp), dimension(:), intent(in) y,
    integer(i4), intent(in) k,
    logical, dimension(:), intent(in), optional mask ) [private]
```

15.17.1.17 crosscorr_sp()

```
real(sp) function mo_corr::crosscorr_sp (
    real(sp), dimension(:), intent(in) x,
    real(sp), dimension(:), intent(in) y,
    integer(i4), intent(in) k,
    logical, dimension(:), intent(in), optional mask ) [private]
```

15.17.1.18 four1_dp()

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

References `mo_kind::dpc`, and `mo_constants::twopi_dp`.

15.17.1.19 four1_sp()

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

References `mo_kind::dpc`, `mo_kind::spc`, and `mo_constants::twopi_dp`.

15.17.1.20 fourrow_dp()

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

References mo_kind::dpc, and mo_constants::pi_dp.

15.17.1.21 fourrow_sp()

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

References mo_kind::dpc, mo_constants::pi_dp, and mo_kind::spc.

15.17.1.22 realft_dp()

```
subroutine mo_corr::realft_dp (
    real(dp), dimension(:, ), intent(inout) data,
    integer(i4), intent(in) isign,
    complex(dpc), dimension(:, ), optional, target zdata ) [private]
```

References mo_kind::dpc, and zroots_unity_dp().

Here is the call graph for this function:

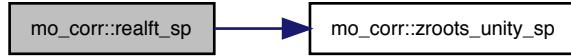


15.17.1.23 realft_sp()

```
subroutine mo_corr::realft_sp (
    real(spc), dimension(:, ), intent(inout) data,
    integer(i4), intent(in) isign,
    complex(spc), dimension(:, ), optional, target zdata ) [private]
```

References mo_kind::spc, and zroots_unity_sp().

Here is the call graph for this function:



15.17.1.24 swap_1d_dpc()

```
subroutine mo_corr::swap_1d_dpc (
    complex(dpc), dimension(:), intent(inout) a,
    complex(dpc), dimension(:), intent(inout) b ) [private]
```

15.17.1.25 swap_1d_spc()

```
subroutine mo_corr::swap_1d_spc (
    complex(spc), dimension(:), intent(inout) a,
    complex(spc), dimension(:), intent(inout) b ) [private]
```

15.17.1.26 zroots_unity_dp()

```
complex(dpc) function, dimension(nn) mo_corr::zroots_unity_dp (
    integer(i4), intent(in) n,
    integer(i4), intent(in) nn ) [private]
```

References mo_kind::dpc, and mo_constants::twopi_dp.

Referenced by realft_dp().

Here is the caller graph for this function:



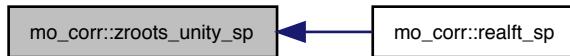
15.17.1.27 zroots_unity_sp()

```
complex(spc) function, dimension(nn) mo_corr::zroots_unity_sp (
    integer(i4), intent(in) n,
    integer(i4), intent(in) nn ) [private]
```

References mo_kind::spc, and mo_constants::twopi_sp.

Referenced by realft_sp().

Here is the caller graph for this function:



15.17.2 Variable Documentation

15.17.2.1 npar2_arth

```
integer(i4), parameter mo_corr::npar2_arth = 8 [private]
```

Referenced by arth_dp(), arth_i4(), and arth_sp().

15.17.2.2 npar_arth

```
integer(i4), parameter mo_corr::npar_arth = 16 [private]
```

Referenced by arth_dp(), arth_i4(), and arth_sp().

15.18 mo_dds Module Reference

Dynamically Dimensioned Search (DDS)

Functions/Subroutines

- real(dp) function, dimension(size(pini)), public **dds** (eval, obj_func, pini, prange, r, seed, maxiter, maxit, mask, tmp_file, funcbest, history)

DDS.
- real(dp) function, dimension(size(pini)), public **mdds** (eval, obj_func, pini, prange, seed, maxiter, maxit, mask, tmp_file, funcbest, history)

MDDS.
- subroutine **neigh_value** (x_cur, x_min, x_max, r, new_value)

15.18.1 Detailed Description

Dynamically Dimensioned Search (DDS)

This module provides routines for Dynamically Dimensioned Search (DDS) of Tolson and Shoemaker (2007). It searches the minimum or maximum of a user-specified function, using an n-dimensional continuous global optimization algorithm (DDS).

Authors

Original by Bryan Tolson and later modified by Rohini Kumar. Matthias Cuntz and Juliane Mai for the module, MDDS, etc.

Date

Jul 2012

15.18.2 Function/Subroutine Documentation

15.18.2.1 dds()

```
real(dp) function, dimension(size(pini)), public mo_dds::dds (
    procedure(eval_interface), intent(in), pointer eval,
    procedure(objective_interface), intent(in), pointer obj_func,
    real(dp), dimension(:), intent(in) pini,
    real(dp), dimension(:, :), intent(in) prange,
    real(dp), intent(in), optional r,
    integer(i8), intent(in), optional seed,
    integer(i8), intent(in), optional maxiter,
    logical, intent(in), optional maxit,
    logical, dimension(:), intent(in), optional mask,
    character(len = *), intent(in), optional tmp_file,
    real(dp), intent(out), optional funcbest,
    real(dp), dimension(:, ), intent(out), optional, allocatable history )
```

DDS.

Searches Minimum or Maximum of a user-specified function using Dynamically Dimensioned Search (DDS).

DDS is an n-dimensional continuous global optimization algorithm. It is coded as a minimizer but one can give maxit=True in a maximization problem, so that the algorithm minimizes the negative of the objective function $F=(-1*F)$. The function to be minimized is the first argument of DDS and must be defined as

```
function func(p)
use mo_kind, only: dp
implicit none
real(dp), dimension(:), intent(in) :: p
real(dp) :: func
end function func
```

Parameters

in	<code>real(dp) :: obj_func(p)</code>	Function on which to search the minimum
in	<code>real(dp) :: pini(:)</code>	initial value of decision variables
in	<code>real(dp) :: prange(size(pini),2)</code>	Min/max range of decision variables
in	<code>real(dp), optional :: r</code>	DDS perturbation parameter (default: 0.2)

Parameters

in	<i>integer(i8), optional :: seed</i>	User seed to initialise the random number generator (default: None)
in	<i>integer(i8), optional :: maxiter</i>	Maximum number of iteration or function evaluation (default: 1000)
in	<i>logical, optional :: maxit</i>	Maximization (.True.) or minimization (.False.) of function (default: .False.)
in	<i>logical, optional :: mask(size(pini))</i>	parameter to be optimized (true or false) (default: .True.)
in	<i>character(len=*) , optional :: tmp_file</i>	file with temporal output
out	<i>real(dp), optional :: funcbest</i>	the best value of the function.
out	<i>real(dp), optional, allocatable :: history(:)</i>	the history of best function values, history(maxiter)=funcbest allocatable only to be in correspondance with other optimization routines

Returns

real(dp) :: DDS — The parameters of the point which is estimated to minimize the function.

Author

Written original Bryan Tolson - DDS v1.1

Modified Rohini Kumar, Matthias Cuntz, Juliane Mai

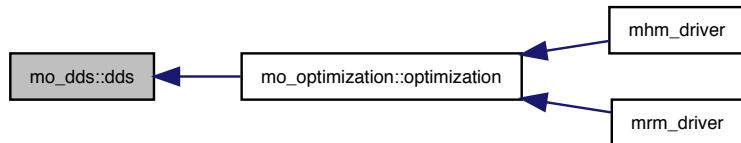
References *mo_kind::dp*, *mo_kind::i4*, *mo_kind::i8*, and *neigh_value()*.

Referenced by *mo_optimization::optimization()*.

Here is the call graph for this function:



Here is the caller graph for this function:



15.18.2.2 mdds()

```
real(dp) function, dimension(size(pini)), public mo_dds::mdds (
    procedure(eval_interface), intent(in), pointer eval,
    procedure(objective_interface), intent(in), pointer obj_func,
    real(dp), dimension(:), intent(in) pini,
    real(dp), dimension(:, :), intent(in) prange,
    integer(i8), intent(in), optional seed,
    integer(i8), intent(in), optional maxiter,
    logical, intent(in), optional maxit,
    logical, dimension(:), intent(in), optional mask,
    character(len = *), intent(in), optional tmp_file,
    real(dp), intent(out), optional funcbest,
    real(dp), dimension(:), intent(out), optional, allocatable history )
```

MDDS.

Searches Minimum or Maximum of a user-specified function using the Modified Dynamically Dimensioned Search (DDS). DDS is an n-dimensional continuous global optimization algorithm. It is coded as a minimizer but one can give maxit=True in a maximization problem, so that the algorithm minimizes the negative of the objective function $F=(-1*F)$.

The function to be minimized is the first argument of DDS and must be defined as

```
function func(p)
  use mo_kind, only: dp
  implicit none
  real(dp), dimension(:), intent(in) :: p
  real(dp) :: func
end function func
```

MDDS extends normal DDS by a continuous reduction of the DDS perturbation parameter r from 0.3 to 0.05, and by allowing a larger function value with a certain probability.

Parameters

in	<i>real(dp) :: obj_func(p)</i>	Function on which to search the minimum
in	<i>real(dp) :: pini(:)</i>	Initial value of decision variables
in	<i>real(dp) :: prange(size(pini),2)</i>	Min/max range of decision variables
in	<i>integer(i8), optional :: seed</i>	User seed to initialise the random number generator (default: None)
in	<i>integer(i8), optional :: maxiter</i>	Maximum number of iteration or function evaluation (default: 1000)
in	<i>logical, optional :: maxit</i>	Maximization (.True.) or minimization (.False.) of function (default: .False.)
in	<i>logical, optional :: mask(size(pini))</i>	Parameter to be optimized (true or false) (default: .True.)
in	<i>character(len=*) , optional :: tmp_file</i>	file with temporal output
out	<i>real(dp), optional :: funcbest</i>	the best value of the function.
out	<i>real(dp), optional, allocatable :: history(:)</i>	the history of best function values,

Returns

real(dp) :: MDDS — The parameters of the point which is estimated to minimize the function.

Author

Written Matthias Cuntz and Juliane Mai

References `mo_kind::dp`, `mo_kind::i4`, `mo_kind::i8`, and `neigh_value()`.

Here is the call graph for this function:



15.18.2.3 `neigh_value()`

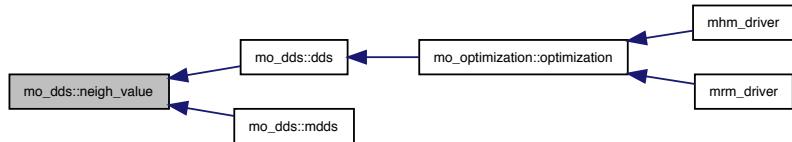
```

subroutine mo_dds::neigh_value (
    real(dp), intent(in) x_cur,
    real(dp), intent(in) x_min,
    real(dp), intent(in) x_max,
    real(dp), intent(in) r,
    real(dp), intent(out) new_value )
  
```

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

Referenced by `dds()`, and `mdds()`.

Here is the caller graph for this function:



15.19 `mo_errormeasures` Module Reference

Data Types

- interface [bias](#)
- interface [kge](#)
Kling-Gupta-Efficiency measure.
- interface [kgenocorr](#)
Kling-Gupta-Efficiency measure without correlation.
- interface [Innse](#)
- interface [mae](#)

- interface `mse`
- interface `nse`
- interface `rmse`
- interface `sae`
- interface `sse`
- interface `wnse`

Functions/Subroutines

- real(sp) function `bias_sp_1d` (x, y, mask)
- real(dp) function `bias_dp_1d` (x, y, mask)
- real(sp) function `bias_sp_2d` (x, y, mask)
- real(dp) function `bias_dp_2d` (x, y, mask)
- real(sp) function `bias_sp_3d` (x, y, mask)
- real(dp) function `bias_dp_3d` (x, y, mask)
- real(sp) function `kge_sp_1d` (x, y, mask)
- real(sp) function `kge_sp_2d` (x, y, mask)
- real(sp) function `kge_sp_3d` (x, y, mask)
- real(dp) function `kge_dp_1d` (x, y, mask)
- real(dp) function `kge_dp_2d` (x, y, mask)
- real(dp) function `kge_dp_3d` (x, y, mask)
- real(sp) function `kgenocorr_sp_1d` (x, y, mask)
- real(sp) function `kgenocorr_sp_2d` (x, y, mask)
- real(sp) function `kgenocorr_sp_3d` (x, y, mask)
- real(dp) function `kgenocorr_dp_1d` (x, y, mask)
- real(dp) function `kgenocorr_dp_2d` (x, y, mask)
- real(dp) function `kgenocorr_dp_3d` (x, y, mask)
- real(sp) function `lnnse_sp_1d` (x, y, mask)
- real(dp) function `lnnse_dp_1d` (x, y, mask)
- real(sp) function `lnnse_sp_2d` (x, y, mask)
- real(dp) function `lnnse_dp_2d` (x, y, mask)
- real(sp) function `lnnse_sp_3d` (x, y, mask)
- real(dp) function `lnnse_dp_3d` (x, y, mask)
- real(sp) function `mae_sp_1d` (x, y, mask)
- real(dp) function `mae_dp_1d` (x, y, mask)
- real(sp) function `mae_sp_2d` (x, y, mask)
- real(dp) function `mae_dp_2d` (x, y, mask)
- real(sp) function `mae_sp_3d` (x, y, mask)
- real(dp) function `mae_dp_3d` (x, y, mask)
- real(sp) function `mse_sp_1d` (x, y, mask)
- real(dp) function `mse_dp_1d` (x, y, mask)
- real(sp) function `mse_sp_2d` (x, y, mask)
- real(dp) function `mse_dp_2d` (x, y, mask)
- real(sp) function `mse_sp_3d` (x, y, mask)
- real(dp) function `mse_dp_3d` (x, y, mask)
- real(sp) function `nse_sp_1d` (x, y, mask)
- real(dp) function `nse_dp_1d` (x, y, mask)
- real(sp) function `nse_sp_2d` (x, y, mask)
- real(dp) function `nse_dp_2d` (x, y, mask)
- real(sp) function `nse_sp_3d` (x, y, mask)
- real(dp) function `nse_dp_3d` (x, y, mask)
- real(sp) function `sae_sp_1d` (x, y, mask)
- real(dp) function `sae_dp_1d` (x, y, mask)
- real(sp) function `sae_sp_2d` (x, y, mask)

- real(dp) function `sae_dp_2d` (x, y, mask)
- real(sp) function `sae_sp_3d` (x, y, mask)
- real(dp) function `sae_dp_3d` (x, y, mask)
- real(sp) function `sse_sp_1d` (x, y, mask)
- real(dp) function `sse_dp_1d` (x, y, mask)
- real(sp) function `sse_sp_2d` (x, y, mask)
- real(dp) function `sse_dp_2d` (x, y, mask)
- real(sp) function `sse_sp_3d` (x, y, mask)
- real(dp) function `sse_dp_3d` (x, y, mask)
- real(sp) function `rmse_sp_1d` (x, y, mask)
- real(dp) function `rmse_dp_1d` (x, y, mask)
- real(sp) function `rmse_sp_2d` (x, y, mask)
- real(dp) function `rmse_dp_2d` (x, y, mask)
- real(sp) function `rmse_sp_3d` (x, y, mask)
- real(dp) function `rmse_dp_3d` (x, y, mask)
- real(sp) function `wnse_sp_1d` (x, y, mask)
- real(dp) function `wnse_dp_1d` (x, y, mask)
- real(sp) function `wnse_sp_2d` (x, y, mask)
- real(dp) function `wnse_dp_2d` (x, y, mask)
- real(sp) function `wnse_sp_3d` (x, y, mask)
- real(dp) function `wnse_dp_3d` (x, y, mask)

15.19.1 Function/Subroutine Documentation

15.19.1.1 `bias_dp_1d()`

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

15.19.1.2 `bias_dp_2d()`

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

15.19.1.3 `bias_dp_3d()`

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

15.19.1.4 bias_sp_1d()

```
real(sp) function mo_errormeasures::bias_sp_1d (
    real(sp), dimension(:), intent(in) x,
    real(sp), dimension(:), intent(in) y,
    logical, dimension(:), intent(in), optional mask ) [private]
```

15.19.1.5 bias_sp_2d()

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

15.19.1.6 bias_sp_3d()

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

15.19.1.7 kge_dp_1d()

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

15.19.1.8 kge_dp_2d()

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

15.19.1.9 kge_dp_3d()

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

15.19.1.10 kge_sp_1d()

```
real(sp) function mo_errormeasures::kge_sp_1d (
```

```
real(sp), dimension(:), intent(in) x,
real(sp), dimension(:), intent(in) y,
logical, dimension(:), intent(in), optional mask )
```

15.19.1.11 kge_sp_2d()

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

15.19.1.12 kge_sp_3d()

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

15.19.1.13 kgenocorr_dp_1d()

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

15.19.1.14 kgenocorr_dp_2d()

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

15.19.1.15 kgenocorr_dp_3d()

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

15.19.1.16 kgenocorr_sp_1d()

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

15.19.1.17 kgenocorr_sp_2d()

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

15.19.1.18 kgenocorr_sp_3d()

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

15.19.1.19 lnnse_dp_1d()

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

15.19.1.20 lnnse_dp_2d()

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

15.19.1.21 lnnse_dp_3d()

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

15.19.1.22 lnnse_sp_1d()

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

15.19.1.23 `lnnse_sp_2d()`

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

15.19.1.24 `lnnse_sp_3d()`

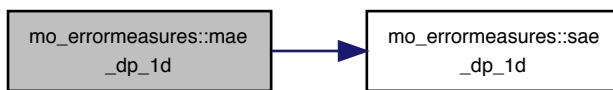
```
real(sp) function mo_errormeasures::lnnse_sp_3d (
    real(sp), dimension(:, :, :, :), intent(in) x,
    real(sp), dimension(:, :, :, :), intent(in) y,
    logical, dimension(:, :, :, :), intent(inout), optional mask )
```

15.19.1.25 `mae_dp_1d()`

```
real(dp) function mo_errormeasures::mae_dp_1d (
    real(dp), dimension(:, :), intent(in) x,
    real(dp), dimension(:, :), intent(in) y,
    logical, dimension(:, :), intent(in), optional mask ) [private]
```

References `sae_dp_1d()`.

Here is the call graph for this function:

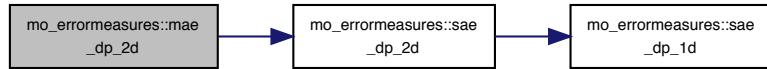


15.19.1.26 `mae_dp_2d()`

```
real(dp) function mo_errormeasures::mae_dp_2d (
    real(dp), dimension(:, :, :), intent(in) x,
    real(dp), dimension(:, :, :), intent(in) y,
    logical, dimension(:, :, :), intent(in), optional mask ) [private]
```

References `sae_dp_2d()`.

Here is the call graph for this function:

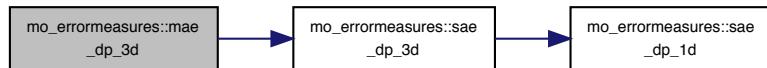


15.19.1.27 mae_dp_3d()

```
real(dp) function mo_errormeasures::mae_dp_3d (
    real(dp), dimension(:, :, :, :), intent(in) x,
    real(dp), dimension(:, :, :, :), intent(in) y,
    logical, dimension(:, :, :, :), intent(in), optional mask ) [private]
```

References sae_dp_3d().

Here is the call graph for this function:

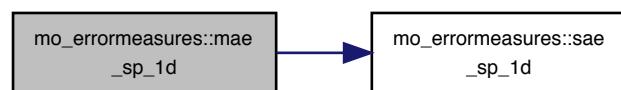


15.19.1.28 mae_sp_1d()

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

References sae_sp_1d().

Here is the call graph for this function:

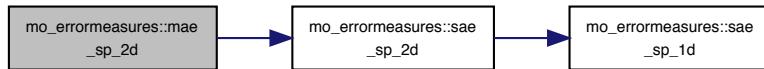


15.19.1.29 mae_sp_2d()

```
real(sp) function mo_errormeasures::mae_sp_2d (
    real(sp), dimension(:, :, ), intent(in) x,
    real(sp), dimension(:, :, ), intent(in) y,
    logical, dimension(:, :, ), intent(in), optional mask ) [private]
```

References sae_sp_2d().

Here is the call graph for this function:

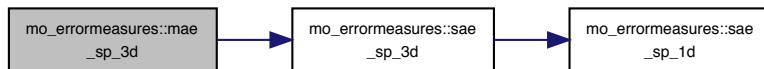


15.19.1.30 mae_sp_3d()

```
real(sp) function mo_errormeasures::mae_sp_3d (
    real(sp), dimension(:, :, :, ), intent(in) x,
    real(sp), dimension(:, :, :, ), intent(in) y,
    logical, dimension(:, :, :, ), intent(in), optional mask ) [private]
```

References sae_sp_3d().

Here is the call graph for this function:



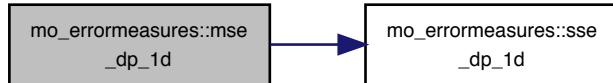
15.19.1.31 mse_dp_1d()

```
real(dp) function mo_errormeasures::mse_dp_1d (
    real(dp), dimension(:, ), intent(in) x,
    real(dp), dimension(:, ), intent(in) y,
    logical, dimension(:, ), intent(in), optional mask ) [private]
```

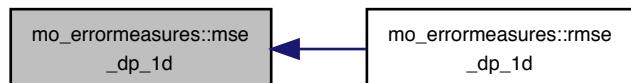
References sse_dp_1d().

Referenced by rmse_dp_1d().

Here is the call graph for this function:



Here is the caller graph for this function:



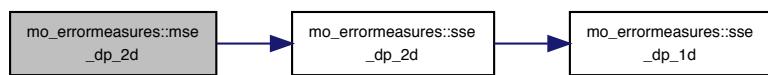
15.19.1.32 mse_dp_2d()

```
real(dp) function mo_errormeasures::mse_dp_2d (  
    real(dp), dimension(:, :, ), intent(in) x,  
    real(dp), dimension(:, :, ), intent(in) y,  
    logical, dimension(:, :, ), intent(in), optional mask ) [private]
```

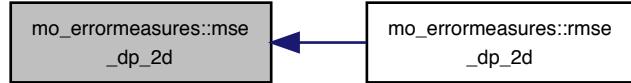
References sse_dp_2d().

Referenced by rmse_dp_2d().

Here is the call graph for this function:



Here is the caller graph for this function:



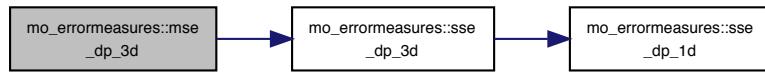
15.19.1.33 mse_dp_3d()

```
real(dp) function mo_errormeasures::mse_dp_3d (
    real(dp), dimension(:, :, :), intent(in) x,
    real(dp), dimension(:, :, :), intent(in) y,
    logical, dimension(:, :, :), intent(in), optional mask ) [private]
```

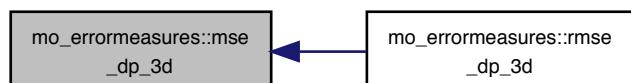
References sse_dp_3d().

Referenced by rmse_dp_3d().

Here is the call graph for this function:



Here is the caller graph for this function:



15.19.1.34 mse_sp_1d()

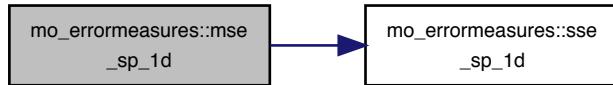
```
real(sp) function mo_errormeasures::mse_sp_1d (
    real(sp), dimension(:), intent(in) x,
```

```
real(sp), dimension(:), intent(in) y,
logical, dimension(:), intent(in), optional mask ) [private]
```

References sse_sp_1d().

Referenced by rmse_sp_1d().

Here is the call graph for this function:



Here is the caller graph for this function:



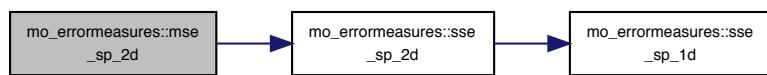
15.19.1.35 mse_sp_2d()

```
real(sp) function mo_errormeasures::mse_sp_2d (
    real(sp), dimension(:, :, :), intent(in) x,
    real(sp), dimension(:, :, :), intent(in) y,
    logical, dimension(:, :, :), intent(in), optional mask ) [private]
```

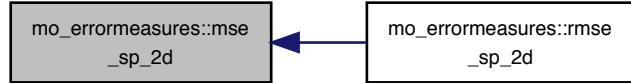
References sse_sp_2d().

Referenced by rmse_sp_2d().

Here is the call graph for this function:



Here is the caller graph for this function:



15.19.1.36 mse_sp_3d()

```

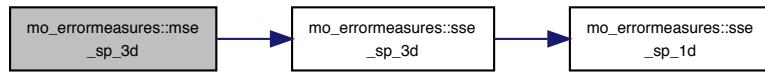
real(sp) function mo_errormeasures::mse_sp_3d (
    real(sp), dimension(:, :, :), intent(in) x,
    real(sp), dimension(:, :, :), intent(in) y,
    logical, dimension(:, :, :), intent(in), optional mask ) [private]

```

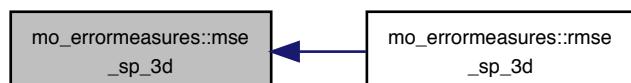
References `sse_sp_3d()`.

Referenced by `rmse_sp_3d()`.

Here is the call graph for this function:



Here is the caller graph for this function:



15.19.1.37 nse_dp_1d()

```

real(dp) function mo_errormeasures::nse_dp_1d (
    real(dp), dimension(:), intent(in) x,

```

```
real(dp), dimension(:), intent(in) y,
logical, dimension(:), intent(in), optional mask )
```

15.19.1.38 nse_dp_2d()

```
real(dp) function mo_errormeasures::nse_dp_2d (
    real(dp), dimension(:, :, ), intent(in) x,
    real(dp), dimension(:, :, ), intent(in) y,
    logical, dimension(:, :, ), intent(in), optional mask )
```

15.19.1.39 nse_dp_3d()

```
real(dp) function mo_errormeasures::nse_dp_3d (
    real(dp), dimension(:, :, :, ), intent(in) x,
    real(dp), dimension(:, :, :, ), intent(in) y,
    logical, dimension(:, :, :, ), intent(in), optional mask )
```

15.19.1.40 nse_sp_1d()

```
real(sp) function mo_errormeasures::nse_sp_1d (
    real(sp), dimension(:, ), intent(in) x,
    real(sp), dimension(:, ), intent(in) y,
    logical, dimension(:, ), intent(in), optional mask ) [private]
```

15.19.1.41 nse_sp_2d()

```
real(sp) function mo_errormeasures::nse_sp_2d (
    real(sp), dimension(:, :, ), intent(in) x,
    real(sp), dimension(:, :, ), intent(in) y,
    logical, dimension(:, :, ), intent(in), optional mask )
```

15.19.1.42 nse_sp_3d()

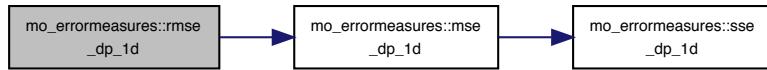
```
real(sp) function mo_errormeasures::nse_sp_3d (
    real(sp), dimension(:, :, :, ), intent(in) x,
    real(sp), dimension(:, :, :, ), intent(in) y,
    logical, dimension(:, :, :, ), intent(in), optional mask )
```

15.19.1.43 rmse_dp_1d()

```
real(dp) function mo_errormeasures::rmse_dp_1d (
    real(dp), dimension(:, ), intent(in) x,
    real(dp), dimension(:, ), intent(in) y,
    logical, dimension(:, ), intent(in), optional mask ) [private]
```

References `mse_dp_1d()`.

Here is the call graph for this function:

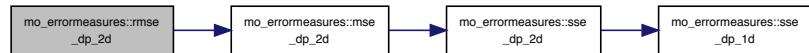


15.19.1.44 `rmse_dp_2d()`

```
real(dp) function mo_errormeasures::rmse_dp_2d (
    real(dp), dimension(:, :, ), intent(in) x,
    real(dp), dimension(:, :, ), intent(in) y,
    logical, dimension(:, :, ), intent(in), optional mask ) [private]
```

References `mse_dp_2d()`.

Here is the call graph for this function:



15.19.1.45 `rmse_dp_3d()`

```
real(dp) function mo_errormeasures::rmse_dp_3d (
    real(dp), dimension(:, :, :, ), intent(in) x,
    real(dp), dimension(:, :, :, ), intent(in) y,
    logical, dimension(:, :, :, ), intent(in), optional mask ) [private]
```

References `mse_dp_3d()`.

Here is the call graph for this function:

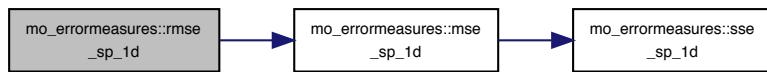


15.19.1.46 rmse_sp_1d()

```
real(sp) function mo_errormeasures::rmse_sp_1d (
    real(sp), dimension(:), intent(in) x,
    real(sp), dimension(:), intent(in) y,
    logical, dimension(:), intent(in), optional mask ) [private]
```

References mse_sp_1d().

Here is the call graph for this function:

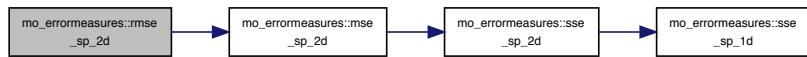


15.19.1.47 rmse_sp_2d()

```
real(sp) function mo_errormeasures::rmse_sp_2d (
    real(sp), dimension(:, :), intent(in) x,
    real(sp), dimension(:, :), intent(in) y,
    logical, dimension(:, :, :), intent(in), optional mask ) [private]
```

References mse_sp_2d().

Here is the call graph for this function:



15.19.1.48 rmse_sp_3d()

```
real(sp) function mo_errormeasures::rmse_sp_3d (
    real(sp), dimension(:, :, :), intent(in) x,
    real(sp), dimension(:, :, :), intent(in) y,
    logical, dimension(:, :, :, :), intent(in), optional mask ) [private]
```

References mse_sp_3d().

Here is the call graph for this function:

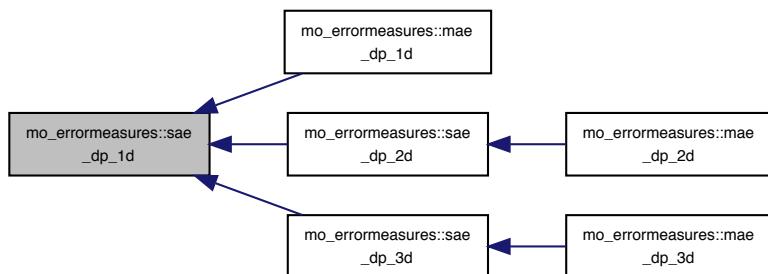


15.19.1.49 sae_dp_1d()

```
real(dp) function mo_errormeasures::sae_dp_1d (
    real(dp), dimension(:), intent(in) x,
    real(dp), dimension(:), intent(in) y,
    logical, dimension(:), intent(in), optional mask ) [private]
```

Referenced by mae_dp_1d(), sae_dp_2d(), and sae_dp_3d().

Here is the caller graph for this function:



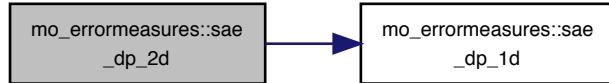
15.19.1.50 sae_dp_2d()

```
real(dp) function mo_errormeasures::sae_dp_2d (
    real(dp), dimension(:, :), intent(in) x,
    real(dp), dimension(:, :), intent(in) y,
    logical, dimension(:, :), intent(in), optional mask ) [private]
```

References sae_dp_1d().

Referenced by mae_dp_2d().

Here is the call graph for this function:



Here is the caller graph for this function:



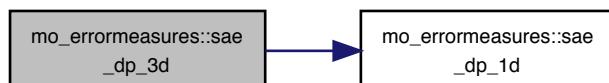
15.19.1.51 sae_dp_3d()

```
real(dp) function mo_errormeasures::sae_dp_3d (
    real(dp), dimension(:, :, :), intent(in) x,
    real(dp), dimension(:, :, :), intent(in) y,
    logical, dimension(:, :, :), intent(in), optional mask ) [private]
```

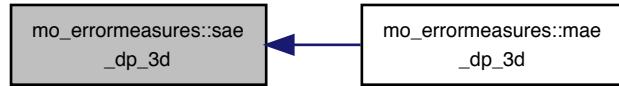
References sae_dp_1d().

Referenced by mae_dp_3d().

Here is the call graph for this function:



Here is the caller graph for this function:



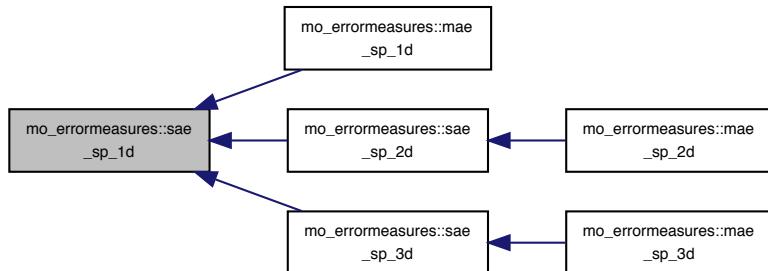
15.19.1.52 sae_sp_1d()

```

real(sp) function mo_errormeasures::sae_sp_1d (
    real(sp), dimension(:), intent(in) x,
    real(sp), dimension(:), intent(in) y,
    logical, dimension(:), intent(in), optional mask )
  
```

Referenced by mae_sp_1d(), sae_sp_2d(), and sae_sp_3d().

Here is the caller graph for this function:



15.19.1.53 sae_sp_2d()

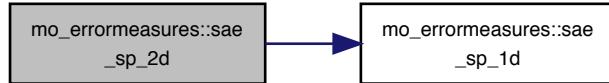
```

real(sp) function mo_errormeasures::sae_sp_2d (
    real(sp), dimension(:, :), intent(in) x,
    real(sp), dimension(:, :), intent(in) y,
    logical, dimension(:, :), intent(in), optional mask ) [private]
  
```

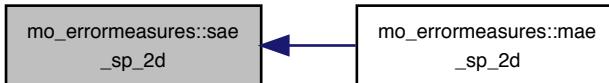
References sae_sp_1d().

Referenced by mae_sp_2d().

Here is the call graph for this function:



Here is the caller graph for this function:



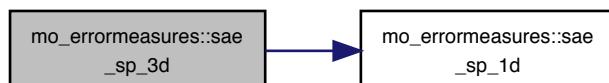
15.19.1.54 sae_sp_3d()

```
real(sp) function mo_errormeasures::sae_sp_3d (
    real(sp), dimension(:, :, :), intent(in) x,
    real(sp), dimension(:, :, :), intent(in) y,
    logical, dimension(:, :, :), intent(in), optional mask ) [private]
```

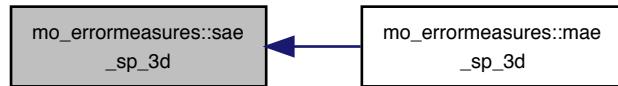
References sae_sp_1d().

Referenced by mae_sp_3d().

Here is the call graph for this function:



Here is the caller graph for this function:

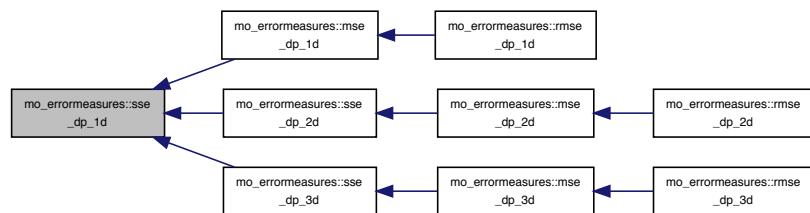


15.19.1.55 sse_dp_1d()

```
real(dp) function mo_errormeasures::sse_dp_1d (
    real(dp), dimension(:), intent(in) x,
    real(dp), dimension(:), intent(in) y,
    logical, dimension(:), intent(in), optional mask ) [private]
```

Referenced by mse_dp_1d(), sse_dp_2d(), and sse_dp_3d().

Here is the caller graph for this function:



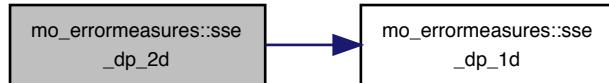
15.19.1.56 sse_dp_2d()

```
real(dp) function mo_errormeasures::sse_dp_2d (
    real(dp), dimension(:, :), intent(in) x,
    real(dp), dimension(:, :), intent(in) y,
    logical, dimension(:, :), intent(in), optional mask ) [private]
```

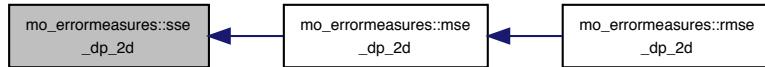
References sse_dp_1d().

Referenced by mse_dp_2d().

Here is the call graph for this function:



Here is the caller graph for this function:



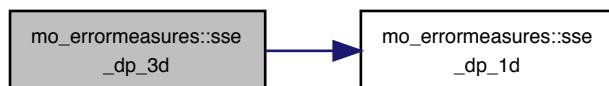
15.19.1.57 sse_dp_3d()

```
real(dp) function mo_errormeasures::sse_dp_3d (
    real(dp), dimension(:, :, :), intent(in) x,
    real(dp), dimension(:, :, :), intent(in) y,
    logical, dimension(:, :, :), intent(in), optional mask ) [private]
```

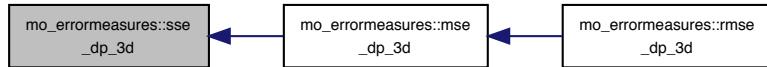
References sse_dp_1d().

Referenced by mse_dp_3d().

Here is the call graph for this function:



Here is the caller graph for this function:

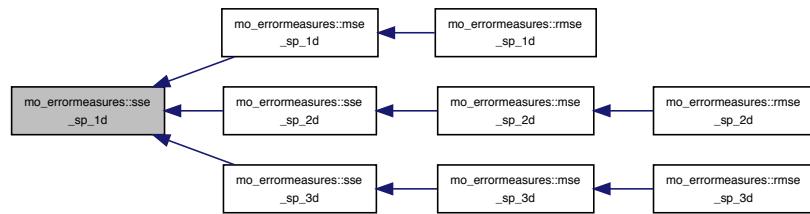


15.19.1.58 sse_sp_1d()

```
real(sp) function mo_errormeasures::sse_sp_1d (
    real(sp), dimension(:), intent(in) x,
    real(sp), dimension(:), intent(in) y,
    logical, dimension(:), intent(in), optional mask ) [private]
```

Referenced by mse_sp_1d(), sse_sp_2d(), and sse_sp_3d().

Here is the caller graph for this function:



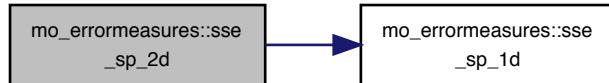
15.19.1.59 sse_sp_2d()

```
real(sp) function mo_errormeasures::sse_sp_2d (
    real(sp), dimension(:, :), intent(in) x,
    real(sp), dimension(:, :), intent(in) y,
    logical, dimension(:, :), intent(in), optional mask ) [private]
```

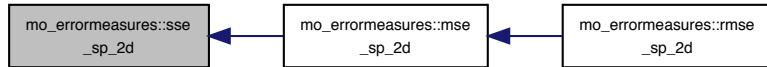
References sse_sp_1d().

Referenced by mse_sp_2d().

Here is the call graph for this function:



Here is the caller graph for this function:



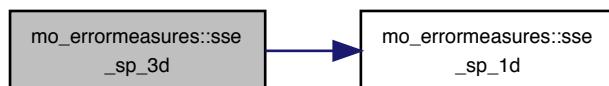
15.19.1.60 sse_sp_3d()

```
real(sp) function mo_errormeasures::sse_sp_3d (
    real(sp), dimension(:, :, :), intent(in) x,
    real(sp), dimension(:, :, :), intent(in) y,
    logical, dimension(:, :, :), intent(in), optional mask ) [private]
```

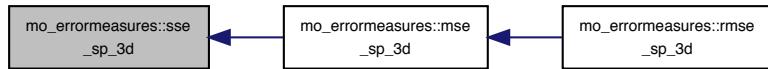
References [sse_sp_1d\(\)](#).

Referenced by [mse_sp_3d\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



15.19.1.61 `wnse_dp_1d()`

```
real(dp) function mo_errormeasures::wnse_dp_1d (
    real(dp), dimension(:), intent(in) x,
    real(dp), dimension(:), intent(in) y,
    logical, dimension(:), intent(in), optional mask )
```

15.19.1.62 `wnse_dp_2d()`

```
real(dp) function mo_errormeasures::wnse_dp_2d (
    real(dp), dimension(:, :, ), intent(in) x,
    real(dp), dimension(:, :, ), intent(in) y,
    logical, dimension(:, :, ), intent(in), optional mask )
```

15.19.1.63 `wnse_dp_3d()`

```
real(dp) function mo_errormeasures::wnse_dp_3d (
    real(dp), dimension(:, :, :, ), intent(in) x,
    real(dp), dimension(:, :, :, ), intent(in) y,
    logical, dimension(:, :, :, ), intent(in), optional mask )
```

15.19.1.64 `wnse_sp_1d()`

```
real(sp) function mo_errormeasures::wnse_sp_1d (
    real(sp), dimension(:), intent(in) x,
    real(sp), dimension(:), intent(in) y,
    logical, dimension(:), intent(in), optional mask ) [private]
```

15.19.1.65 `wnse_sp_2d()`

```
real(sp) function mo_errormeasures::wnse_sp_2d (
    real(sp), dimension(:, :, ), intent(in) x,
    real(sp), dimension(:, :, ), intent(in) y,
    logical, dimension(:, :, ), intent(in), optional mask )
```

15.19.1.66 `wnse_sp_3d()`

```
real(sp) function mo_errormeasures::wnse_sp_3d (
    real(sp), dimension(:,:,:), intent(in) x,
    real(sp), dimension(:,:,:), intent(in) y,
    logical, dimension(:,:,:), intent(in), optional mask )
```

15.20 mo_file Module Reference

Provides file names and units for mHM.

Variables

- 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

15.20.1 Detailed Description

Provides file names and units for mHM.

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

Authors

Matthias Cuntz

Date

Jan 2012

15.20.2 Variable Documentation

15.20.2.1 file_defoutput

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

file defining mHM's outputs

Referenced by mo_mhm_read_config::mhm_read_config().

15.20.2.2 file_main

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

Driver file.

Referenced by mhm_driver().

15.20.2.3 file_namelist_mhm

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

Namelist file name.

15.20.2.4 file_namelist_mhm_param

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

Parameter namelists file name.

Referenced by mo_startup::constants_init().

15.20.2.5 udefoutput

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

Unit for file defining mHM's outputs.

Referenced by mo_mhm_read_config::mhm_read_config().

15.20.2.6 unamelist_mhm

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

Unit for namelist.

15.20.2.7 unamelist_mhm_param

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

Unit for namelist.

15.20.2.8 utws

```
integer, parameter mo_file::utws = 77
unit for tws time series
Referenced by mo_read_optional_data::read_basin_avg_tws().
```

15.20.2.9 version

```
character(len = *), parameter mo_file::version = '5.9'
Current mHM model version.
Referenced by mo_write_fluxes_states::createoutfile(), mhm_driver(), and mo_write_ascii::write_configfile().
```

15.20.2.10 version_date

```
character(len = *), parameter mo_file::version_date = 'June 2018'
Time of current mHM model version release.
Referenced by mhm_driver().
```

15.21 mo_finish Module Reference

Finish a program gracefully.

Functions/Subroutines

- subroutine, public **finish** (name, text, unit)
Finish a program gracefully.

15.21.1 Detailed Description

Finish a program gracefully.

This module supplies a routine that writes out final comments and then stops.

Authors

Original of Echam5, (C) MPI-MET, Hamburg, Germany.
Modified Matthias Cuntz

Date

Jan 2011

15.21.2 Function/Subroutine Documentation

15.21.2.1 `finish()`

```
subroutine, public mo_finish::finish (
    character(len = *), intent(in) name,
    character(len = *), intent(in), optional text,
    integer, intent(in), optional unit )
```

Finish a program gracefully.

Stop a program but writing out a message first that is separated from earlier output by ----- (i.e. the separator of [mo_string_utils](#))

Parameters

in	<code>character(len=*) :: name</code>	First string separated from optional second by :
in	<code>character(len=*), optional :: text</code>	Second string separated by :
in	<code>integer, optional :: unit</code>	File unit for write (default: *)

Author

Written, Matthias Cuntz - modified from Echam5, (C) MPI-MET, Hamburg, Germany

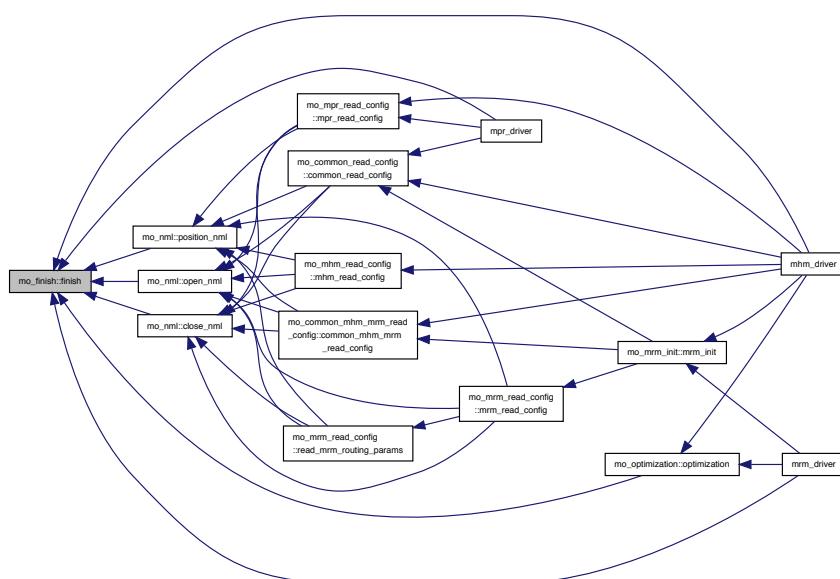
Date

Dec 2011

References `mo_string_utils::separator`.

Referenced by `mo_nml::close_nml()`, `mhm_driver()`, `mpr_driver()`, `mrm_driver()`, `mo_nml::open_nml()`, `mo_optimization::optimization()`, and `mo_nml::position_nml()`.

Here is the caller graph for this function:



15.22 mo_global_variables Module Reference

Global variables ONLY used in reading, writing and startup.

Data Types

- type `twsstructure`

Variables

- integer(i4) `timestep_model_outputs`
- logical, dimension(noutflxstate) `outputflxstate`
- integer(i4), dimension(:,), allocatable, public `timestep_model_inputs`
- logical, public `read_meteo_weights`
- character(256), public `inputformat_meteo_forcings`
- integer(i4), public `timestep_sm_input`
- integer(i4), public `timestep_neutrons_input`
- integer(i4), public `timestep_et_input`
- character(256), dimension(:,), allocatable, public `dirprecipitation`
- character(256), dimension(:,), allocatable, public `dirtemperature`
- character(256), dimension(:,), allocatable, public `dirmintemperature`
- character(256), dimension(:,), allocatable, public `dirmaxtemperature`
- character(256), dimension(:,), allocatable, public `dirnetradiation`
- character(256), dimension(:,), allocatable, public `dirabsvappressure`
- character(256), dimension(:,), allocatable, public `dirwindspeed`
- character(256), dimension(:,), allocatable, public `dirreferenceet`
- character(256), dimension(:,), allocatable, public `dirsoil_moisture`
- character(256), dimension(:,), allocatable, public `filetws`
- character(256), dimension(:,), allocatable, public `dirneutrons`
- character(256), dimension(:,), allocatable, public `direvapotranspiration`
- integer(i4), parameter, public `routingstates` = 2
- type(`twsstructure`), public `basin_avg_tws_obs`
- real(dp), dimension(:, :,), allocatable, public `basin_avg_tws_sim`
- integer(i4), public `nmeasperday_tws`
- type(grid), dimension(:,), allocatable, public `level2`
- real(dp), dimension(:, :, :,), allocatable, public `l1_temp_weights`
- real(dp), dimension(:, :, :,), allocatable, public `l1_pet_weights`
- real(dp), dimension(:, :, :,), allocatable, public `l1_pre_weights`
- real(dp), dimension(:, :,), allocatable, public `l1_pre`
- real(dp), dimension(:, :,), allocatable, public `l1_temp`
- real(dp), dimension(:, :,), allocatable, public `l1_pet`
- real(dp), dimension(:, :,), allocatable, public `l1_tmin`
- real(dp), dimension(:, :,), allocatable, public `l1_tmax`
- real(dp), dimension(:, :,), allocatable, public `l1_netrad`
- real(dp), dimension(:, :,), allocatable, public `l1_absvappress`
- real(dp), dimension(:, :,), allocatable, public `l1_windspeed`
- real(dp), dimension(:, :,), allocatable, public `l1_sm`
- logical, dimension(:, :,), allocatable, public `l1_sm_mask`
- integer(i4) `ntimesteps_l1_sm`
- integer(i4) `nsoilhorizons_sm_input`
- real(dp), dimension(:, :,), allocatable, public `l1_neutronsdata`
- logical, dimension(:, :,), allocatable, public `l1_neutronsdata_mask`
- integer(i4) `ntimesteps_l1_neutrons`

- real(dp), dimension(:, :, :), allocatable, public `l1_et`
- logical, dimension(:, :, :), allocatable, public `l1_et_mask`
- integer(i4) `ntimesteps_l1_et`
- real(dp), dimension(:, :, :), allocatable, public `l1_inter`
- real(dp), dimension(:, :, :), allocatable, public `l1_snowpack`
- real(dp), dimension(:, :, :), allocatable, public `l1_sealstw`
- real(dp), dimension(:, :, :), allocatable, public `l1_soilmoist`
- real(dp), dimension(:, :, :), allocatable, public `l1_unsatstw`
- real(dp), dimension(:, :, :), allocatable, public `l1_satstw`
- real(dp), dimension(:, :, :), allocatable, public `l1_neutrons`
- real(dp), dimension(:, :, :), allocatable, public `l1_pet_calc`
- real(dp), dimension(:, :, :), allocatable, public `l1_aetsoil`
- real(dp), dimension(:, :, :), allocatable, public `l1_aetcanopy`
- real(dp), dimension(:, :, :), allocatable, public `l1_aetsealed`
- real(dp), dimension(:, :, :), allocatable, public `l1_baseflow`
- real(dp), dimension(:, :, :), allocatable, public `l1_infilsoil`
- real(dp), dimension(:, :, :), allocatable, public `l1_fastrunoff`
- real(dp), dimension(:, :, :), allocatable, public `l1_melt`
- real(dp), dimension(:, :, :), allocatable, public `l1_percol`
- real(dp), dimension(:, :, :), allocatable, public `l1_preeffect`
- real(dp), dimension(:, :, :), allocatable, public `l1_rain`
- real(dp), dimension(:, :, :), allocatable, public `l1_runoffseal`
- real(dp), dimension(:, :, :), allocatable, public `l1_slowrunoff`
- real(dp), dimension(:, :, :), allocatable, public `l1_snow`
- real(dp), dimension(:, :, :), allocatable, public `l1_throughfall`
- real(dp), dimension(:, :, :), allocatable, public `l1_total_runoff`
- real(dp), dimension(int(yearmonths, i4)), public `evap_coeff`
- real(dp), dimension(int(yearmonths, i4)), public `fday_prec`
- real(dp), dimension(int(yearmonths, i4)), public `fnight_prec`
- real(dp), dimension(int(yearmonths, i4)), public `fday_pet`
- real(dp), dimension(int(yearmonths, i4)), public `fnight_pet`
- real(dp), dimension(int(yearmonths, i4)), public `fday_temp`
- real(dp), dimension(int(yearmonths, i4)), public `fnight_temp`
- real(dp), dimension(:, :, :), allocatable, public `neutron_integral_afast`

15.22.1 Detailed Description

Global variables ONLY used in reading, writing and startup.

TODO: add description

Authors

Luis Samaniego

Date

Dec 2012

15.22.2 Variable Documentation

15.22.2.1 basin_avg_tws_obs

```
type(twsstructure), public mo_global_variables::basin_avg_tws_obs
```

Referenced by mo_objective_function::extract_basin_avg_tws(), mo_mhm_read_config::mhm_read_config(), and mo_read_optional_data::read_basin_avg_tws().

15.22.2.2 basin_avg_tws_sim

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

Referenced by mo_mhm_eval::mhm_eval(), and mo_read_optional_data::read_basin_avg_tws().

15.22.2.3 dirabsvappressure

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

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

15.22.2.4 direvapotranspiration

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

Referenced by mo_mhm_read_config::mhm_read_config(), and mo_read_optional_data::read_evapotranspiration().

15.22.2.5 dirmaxtemperature

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

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

15.22.2.6 dirmintemperature

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

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

15.22.2.7 dirnetradiation

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

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

15.22.2.8 dirneutrons

character(256), dimension(:), allocatable, public mo_global_variables::dirneutrons

Referenced by mo_mhm_read_config::mhm_read_config(), and mo_read_optional_data::read_neutrons().

15.22.2.9 dirprecipitation

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

Referenced by mo_startup::l2_variable_init(), mhm_driver(), mo_mhm_read_config::mhm_read_config(), mo_meteo_forcings::prepare_meteo_forcings_data(), and mo_write_ascii::write_configfile().

15.22.2.10 dirreferenceet

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

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

15.22.2.11 dirsoil_moisture

character(256), dimension(:), allocatable, public mo_global_variables::dirsoil_moisture

Referenced by mo_mhm_read_config::mhm_read_config(), and mo_read_optional_data::read_soil_moisture().

15.22.2.12 dirtemperature

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

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

15.22.2.13 dirwindspeed

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

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

15.22.2.14 evap_coeff

real(dp), dimension(int(yearmonths, i4)), public mo_global_variables::evap_coeff

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

15.22.2.15 fday_pet

```
real(dp), dimension(int(yearmonths, i4)), public mo_global_variables::fday_pet
```

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

15.22.2.16 fday_prec

```
real(dp), dimension(int(yearmonths, i4)), public mo_global_variables::fday_prec
```

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

15.22.2.17 fday_temp

```
real(dp), dimension(int(yearmonths, i4)), public mo_global_variables::fday_temp
```

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

15.22.2.18 filetws

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

Referenced by mo_mhm_read_config::mhm_read_config().

15.22.2.19 fnight_pet

```
real(dp), dimension(int(yearmonths, i4)), public mo_global_variables::fnight_pet
```

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

15.22.2.20 fnight_prec

```
real(dp), dimension(int(yearmonths, i4)), public mo_global_variables::fnight_prec
```

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

15.22.2.21 fnight_temp

```
real(dp), dimension(int(yearmonths, i4)), public mo_global_variables::fnight_temp
```

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

15.22.2.22 inputformat_meteo_forcings

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

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

15.22.2.23 l1_absvapress

```
real(dp), dimension(:, :, ), allocatable, public mo_global_variables::l1_absvapress
```

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

15.22.2.24 l1_aetc canopy

```
real(dp), dimension(:, ), allocatable, public mo_global_variables::l1_aetc canopy
```

Referenced by `mo_mhm_eval::mhm_eval()`, `mo_restart::read_restart_states()`, `mo_init_states::variables_alloc()`, `mo_init_states::variables_default_init()`, and `mo_restart::write_restart_files()`.

15.22.2.25 l1_aetsealed

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

Referenced by `mo_mhm_eval::mhm_eval()`, `mo_restart::read_restart_states()`, `mo_init_states::variables_alloc()`, `mo_init_states::variables_default_init()`, and `mo_restart::write_restart_files()`.

15.22.2.26 l1_aetsoil

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

Referenced by `mo_mhm_eval::mhm_eval()`, `mo_restart::read_restart_states()`, `mo_init_states::variables_alloc()`, `mo_init_states::variables_default_init()`, and `mo_restart::write_restart_files()`.

15.22.2.27 l1_baseflow

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

Referenced by `mo_mhm_eval::mhm_eval()`, `mo_restart::read_restart_states()`, `mo_init_states::variables_alloc()`, `mo_init_states::variables_default_init()`, and `mo_restart::write_restart_files()`.

15.22.2.28 l1_et

```
real(dp), dimension(:, :, ), allocatable, public mo_global_variables::l1_et
```

Referenced by `mo_objective_function::objective_et_kge_catchment_avg()`, `mo_objective_function::objective_kge_q_et()`, `mo_objective_function::objective_kge_q_rmse_et()`, and `mo_read_optional_data::read_evapotranspiration()`.

15.22.2.29 l1_et_mask

```
logical, dimension(:, :), allocatable, public mo_global_variables::l1_et_mask
```

Referenced by mo_objective_function::objective_et_kge_catchment_avg(), mo_objective_function::objective_kge_q_et(), mo_objective_function::objective_kge_q_rmse_et(), and mo_read_optional_data::read_evapotranspiration().

15.22.2.30 l1_fastrunoff

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

Referenced by mo_mhm_eval::mhm_eval(), mo_restart::read_restart_states(), mo_init_states::variables_alloc(), mo_init_states::variables_default_init(), and mo_restart::write_restart_files().

15.22.2.31 l1_infilsoil

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

Referenced by mo_mhm_eval::mhm_eval(), mo_restart::read_restart_states(), mo_init_states::variables_alloc(), mo_init_states::variables_default_init(), and mo_restart::write_restart_files().

15.22.2.32 l1_inter

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

Referenced by mo_mhm_eval::mhm_eval(), mo_restart::read_restart_states(), mo_init_states::variables_alloc(), mo_init_states::variables_default_init(), and mo_restart::write_restart_files().

15.22.2.33 l1_melt

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

Referenced by mo_mhm_eval::mhm_eval(), mo_restart::read_restart_states(), mo_init_states::variables_alloc(), mo_init_states::variables_default_init(), and mo_restart::write_restart_files().

15.22.2.34 l1_netrad

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

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

15.22.2.35 l1_neutrons

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

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

15.22.2.36 l1_neutronsdata

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

Referenced by `mo_objective_function::objective_neutrons_kge_catchment_avg()`, and `mo_read_optional_data::read_neutrons()`.

15.22.2.37 l1_neutronsdata_mask

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

Referenced by `mo_objective_function::objective_neutrons_kge_catchment_avg()`, and `mo_read_optional_data::read_neutrons()`.

15.22.2.38 l1_percol

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

Referenced by `mo_mhm_eval::mhm_eval()`, `mo_restart::read_restart_states()`, `mo_init_states::variables_alloc()`, `mo_init_states::variables_default_init()`, and `mo_restart::write_restart_files()`.

15.22.2.39 l1_pet

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

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

15.22.2.40 l1_pet_calc

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

Referenced by `mo_mhm_eval::mhm_eval()`, `mo_init_states::variables_alloc()`, and `mo_init_states::variables_default_init()`.

15.22.2.41 l1_pet_weights

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

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

15.22.2.42 l1_pre

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

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

15.22.2.43 l1_pre_weights

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

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

15.22.2.44 l1_preeffect

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

Referenced by `mo_mhm_eval::mhm_eval()`, `mo_restart::read_restart_states()`, `mo_init_states::variables_alloc()`, `mo_init_states::variables_default_init()`, and `mo_restart::write_restart_files()`.

15.22.2.45 l1_rain

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

Referenced by `mo_mhm_eval::mhm_eval()`, `mo_restart::read_restart_states()`, `mo_init_states::variables_alloc()`, `mo_init_states::variables_default_init()`, and `mo_restart::write_restart_files()`.

15.22.2.46 l1_runoffseal

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

Referenced by `mo_mhm_eval::mhm_eval()`, `mo_restart::read_restart_states()`, `mo_init_states::variables_alloc()`, `mo_init_states::variables_default_init()`, and `mo_restart::write_restart_files()`.

15.22.2.47 l1_satstw

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

Referenced by `mo_mhm_eval::mhm_eval()`, `mo_restart::read_restart_states()`, `mo_init_states::variables_alloc()`, `mo_init_states::variables_default_init()`, and `mo_restart::write_restart_files()`.

15.22.2.48 l1_sealstw

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

Referenced by `mo_mhm_eval::mhm_eval()`, `mo_restart::read_restart_states()`, `mo_init_states::variables_alloc()`, `mo_init_states::variables_default_init()`, and `mo_restart::write_restart_files()`.

15.22.2.49 l1_slowrunoff

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

Referenced by `mo_mhm_eval::mhm_eval()`, `mo_restart::read_restart_states()`, `mo_init_states::variables_alloc()`, `mo_init_states::variables_default_init()`, and `mo_restart::write_restart_files()`.

15.22.2.50 l1_sm

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

Referenced by mo_objective_function::objective_kge_q_sm_corr(), mo_objective_function::objective_sm_corr(), mo_objective_function::objective_sm_kge_catchment_avg(), mo_objective_function::objective_sm_pd(), mo_objective_function::objective_sm_sse_standard_score(), and mo_read_optional_data::read_soil_moisture().

15.22.2.51 l1_sm_mask

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

Referenced by mo_objective_function::objective_kge_q_sm_corr(), mo_objective_function::objective_sm_corr(), mo_objective_function::objective_sm_kge_catchment_avg(), mo_objective_function::objective_sm_pd(), mo_objective_function::objective_sm_sse_standard_score(), and mo_read_optional_data::read_soil_moisture().

15.22.2.52 l1_snow

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

Referenced by mo_mhm_eval::mhm_eval(), mo_restart::read_restart_states(), mo_init_states::variables_alloc(), mo_init_states::variables_default_init(), and mo_restart::write_restart_files().

15.22.2.53 l1_snowpack

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

Referenced by mo_mhm_eval::mhm_eval(), mo_restart::read_restart_states(), mo_init_states::variables_alloc(), mo_init_states::variables_default_init(), and mo_restart::write_restart_files().

15.22.2.54 l1_soilmoist

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

Referenced by mo_mhm_eval::mhm_eval(), mo_restart::read_restart_states(), mo_init_states::variables_alloc(), mo_init_states::variables_default_init(), and mo_restart::write_restart_files().

15.22.2.55 l1_temp

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

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

15.22.2.56 l1_temp_weights

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

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

15.22.2.57 l1_throughfall

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

Referenced by mo_mhm_eval::mhm_eval(), mo_restart::read_restart_states(), mo_init_states::variables_alloc(), mo_init_states::variables_default_init(), and mo_restart::write_restart_files().

15.22.2.58 l1_tmax

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

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

15.22.2.59 l1_tmin

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

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

15.22.2.60 l1_total_runoff

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

Referenced by mo_mhm_eval::mhm_eval(), mo_restart::read_restart_states(), mo_init_states::variables_alloc(), mo_init_states::variables_default_init(), and mo_restart::write_restart_files().

15.22.2.61 l1_unsatstw

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

Referenced by mo_mhm_eval::mhm_eval(), mo_restart::read_restart_states(), mo_init_states::variables_alloc(), mo_init_states::variables_default_init(), and mo_restart::write_restart_files().

15.22.2.62 l1_windspeed

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

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

15.22.2.63 level2

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

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

15.22.2.64 neutron_integral_afast

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

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

15.22.2.65 nmeasperday_tws

```
integer(i4), public mo_global_variables::nmeasperday_tws
```

Referenced by `mo_objective_function::extract_basin_avg_tws()`, and `mo_read_optional_data::read_basin_avg_tws()`.

15.22.2.66 nsoilhorizons_sm_input

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

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

15.22.2.67 ntimesteps_l1_et

```
integer(i4) mo_global_variables::ntimesteps_l1_et
```

Referenced by `mo_mhm_eval::mhm_eval()`, and `mo_read_optional_data::read_evapotranspiration()`.

15.22.2.68 ntimesteps_l1_neutrons

```
integer(i4) mo_global_variables::ntimesteps_l1_neutrons
```

Referenced by `mo_mhm_eval::mhm_eval()`, and `mo_read_optional_data::read_neutrons()`.

15.22.2.69 ntimesteps_l1_sm

```
integer(i4) mo_global_variables::ntimesteps_l1_sm
```

Referenced by `mo_mhm_eval::mhm_eval()`, and `mo_read_optional_data::read_soil_moisture()`.

15.22.2.70 outputflxstate

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

Referenced by `mo_mhm_eval::mhm_eval()`, `mo_mhm_read_config::mhm_read_config()`, `mo_write_fluxes_states::newoutputdataset()`, and `mo_write_fluxes_states::updatedataset()`.

15.22.2.71 read_meteo_weights

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

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

15.22.2.72 routingstates

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

15.22.2.73 timestep_et_input

```
integer(i4), public mo_global_variables::timestep_et_input
```

Referenced by mo_mhm_eval::mhm_eval(), mo_mhm_read_config::mhm_read_config(), mo_objective_function::objective_kge_q_rmse_et(), and mo_read_optional_data::read_evapotranspiration().

15.22.2.74 timestep_model_inputs

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

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

15.22.2.75 timestep_model_outputs

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

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

15.22.2.76 timestep_neutrons_input

```
integer(i4), public mo_global_variables::timestep_neutrons_input
```

Referenced by mo_mhm_read_config::mhm_read_config(), and mo_read_optional_data::read_neutrons().

15.22.2.77 timestep_sm_input

```
integer(i4), public mo_global_variables::timestep_sm_input
```

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

15.23 mo_grid Module Reference

TODO: add description.

Functions/Subroutines

- subroutine, public `init_lowres_level` (highres, target_resolution, lowres, highres_lowres_remap)
Level-1 variable initialization.
- subroutine, public `set_basin_indices` (grids)
TODO: add description.
- subroutine, public `l0_grid_setup` (new_grid)
level 0 variable initialization
- subroutine, public `mapcoordinates` (level, y, x)
Generate map coordinates.
- subroutine, public `geocoordinates` (level, lat, lon)
Generate geographic coordinates.
- subroutine `calculate_grid_properties` (nrowsIn, ncolsIn, xllcornerIn, yllcornerIn, cellsizeln, 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.

15.23.1 Detailed Description

TODO: add description.

TODO: add description

Authors

Robert Schweppe

Date

Jun 2018

15.23.2 Function/Subroutine Documentation

15.23.2.1 calculate_grid_properties()

```
subroutine mo_grid::calculate_grid_properties (
    integer(i4), intent(in) nrowsIn,
    integer(i4), intent(in) ncolsIn,
    real(dp), intent(in) xllcornerIn,
    real(dp), intent(in) yllcornerIn,
    real(dp), intent(in) cellsizeln,
    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	<i>integer(i4) :: nrowsIn</i>	no. of rows at an input level
in	<i>integer(i4) :: ncolsIn</i>	no. of cols at an input level
in	<i>real(dp) :: xllcornerIn</i>	xllcorner at an input level
in	<i>real(dp) :: yllcornerIn</i>	yllcorner at an input level
in	<i>real(dp) :: cellsizeIn</i>	cell size at an input level
in	<i>real(dp) :: aimingResolution</i>	resolution of an output level
out	<i>integer(i4) :: nrowsOut</i>	no. of rows at an output level
out	<i>integer(i4) :: ncolsOut</i>	no. of cols at an output level
out	<i>real(dp) :: xllcornerOut</i>	xllcorner at an output level
out	<i>real(dp) :: yllcornerOut</i>	yllcorner at an output level
out	<i>real(dp) :: cellsizeOut</i>	cell size at an output level

Authors

Matthias Zink & Rohini Kumar

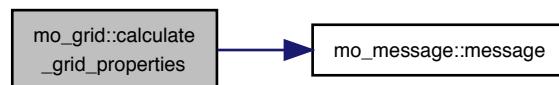
Date

Feb 2013

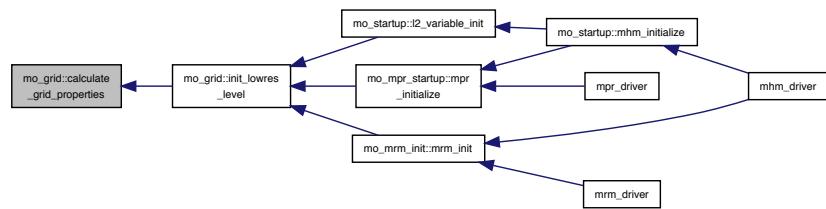
References `mo_message::message()`.

Referenced by `init_lowres_level()`.

Here is the call graph for this function:



Here is the caller graph for this function:



15.23.2.2 geocoordinates()

```
subroutine, public mo_grid::geocoordinates (
    type(grid), intent(in) level,
    real(dp), dimension(:, :), intent(out), allocatable lat,
    real(dp), dimension(:, :), intent(out), allocatable lon )
```

Generate geographic coordinates.

Generate geographic coordinate arrays for given basin and level

Parameters

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

Authors

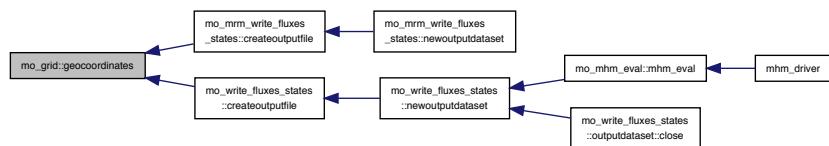
Matthias Zink

Date

Apr 2013

Referenced by `mo_mrm_write_fluxes_states::createoutputfile()`, and `mo_write_fluxes_states::createoutputfile()`.

Here is the caller graph for this function:



15.23.2.3 init_lowres_level()

```
subroutine, public mo_grid::init_lowres_level (
    type(grid), intent(in), target highres,
    real(dp), intent(in) target_resolution,
    type(grid), intent(inout), target lowres,
    type(gridremapper), intent(inout), optional highres_lowres_remap )
```

Level-1 variable initialization.

following tasks are performed for L1 datasets

- cell id & numbering
- mask creation

- storage of cell coordinates (row and column id)
- storage of four corner L0 coordinates 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	<code>type(Grid) :: highres</code>	
in	<code>real(dp) :: target_resolution</code>	
in, out	<code>type(Grid) :: lowres</code>	
in, out	<code>type(GridRemapper), optional :: highres_lowres_remap</code>	

Authors

Rohini Kumar

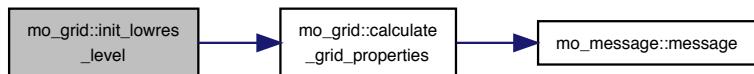
Date

Jan 2013

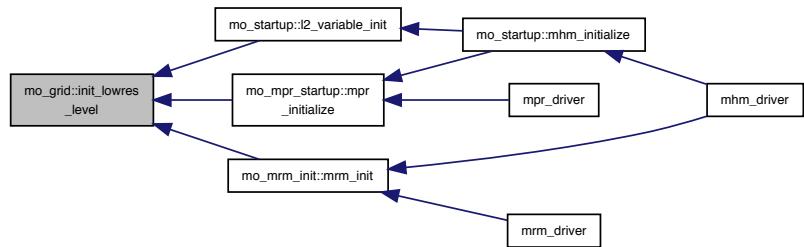
References `calculate_grid_properties()`, `mo_kind::i4`, `mo_common_constants::nodata_dp`, and `mo_common_constants::nodata_i4`.

Referenced by `mo_startup::l2_variable_init()`, `mo_mpr_startup::mpr_initialize()`, and `mo_mrm_init::mrm_init()`.

Here is the call graph for this function:



Here is the caller graph for this function:



15.23.2.4 IO_grid_setup()

```
subroutine, public mo_grid::IO_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 coordinates (row and column id)
- empirical dist. of terrain slope
- flag to determine the presence of a particular soil id in this configuration of the model run If a variable is added or removed here, then it also has to be added or removed in the subroutine config_variables_set in module [mo_restart](#) and in the subroutine set_config in module [mo_set_netcdf_restart](#)

Parameters

in, out	type(Grid) :: new_grid	
---------	------------------------	--

Authors

Rohini Kumar

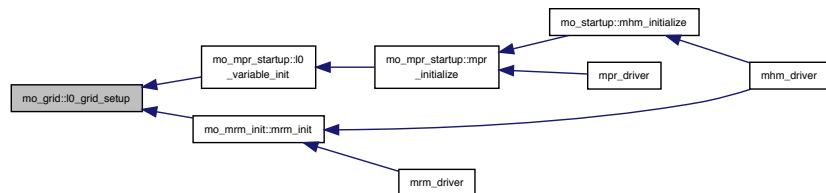
Date

Jan 2013

References `mo_common_variables::iflag_cordinate_sys`, `mo_constants::radiusearth_dp`, and `mo_constants::twopi_dp`.

Referenced by `mo_mpr_startup::IO_variable_init()`, and `mo_mrm_init::mrm_init()`.

Here is the caller graph for this function:



15.23.2.5 mapcoordinates()

```
subroutine, public mo_grid::mapcoordinates (
    type(grid), intent(in) level,
    real(dp), dimension(:), intent(out), allocatable y,
    real(dp), dimension(:), intent(out), allocatable x )
```

Generate map coordinates.

Generate map coordinate arrays for given basin and level

Parameters

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

Authors

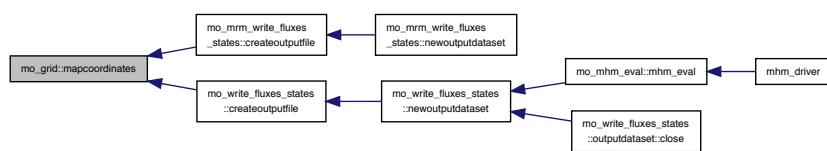
Matthias Zink

Date

Apr 2013

Referenced by `mo_mrm_write_fluxes_states::createoutputfile()`, and `mo_write_fluxes_states::createoutputfile()`.

Here is the caller graph for this function:



15.23.2.6 set_basin_indices()

```
subroutine, public mo_grid::set_basin_indices (
    type(grid), dimension(:), intent(inout) grids )
```

TODO: add description.

TODO: add description

Parameters

in, out	<i>type(Grid), dimension(:) :: grids</i>	
---------	--	--

Authors

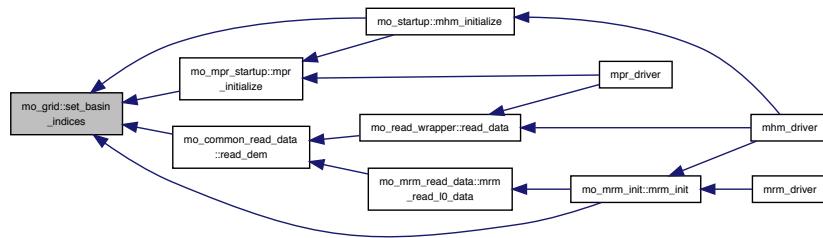
Robert Schweppé

Date

Jun 2018

Referenced by `mo_startup::mhm_initialize()`, `mo_mpr_startup::mpr_initialize()`, `mo_mrm_init::mrm_init()`, and `mo_common_read_data::read_dem()`.

Here is the caller graph for this function:



15.24 mo_init_states Module Reference

Initialization of all state variables of mHM.

Functions/Subroutines

- subroutine, public `variables_alloc` (ncells1)
Allocation of space for mHM related L1 and L11 variables.
- subroutine, public `variables_default_init`
Default initialization mHM related L1 variables.

15.24.1 Detailed Description

Initialization of all state variables of mHM.

This module initializes all state variables required to run mHM. Two options are provided:

- (1) default values
- (2) from nc file

Authors

Luis Samaniego & Rohini Kumar

Date

Dec 2012

15.24.2 Function/Subroutine Documentation

15.24.2.1 variables_alloc()

```
subroutine, public mo_init_states::variables_alloc (
    integer(i4), intent(in) ncells1 )
```

Allocation of space for mHM related L1 and L11 variables.

Allocation of space for mHM related L1 and L11 variables (e.g., states, fluxes, and parameters) for a given basin. Variables allocated here is defined in them [mo_global_variables.f90](#) file. After allocating any variable in this routine, initialize them in the following variables_default_init subroutine:

Parameters

in	integer(i4) :: ncells1	
----	------------------------	--

Authors

Rohini Kumar

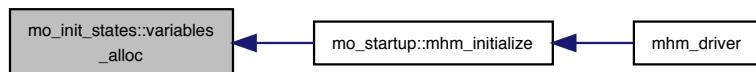
Date

Jan 2013

References mo_mpr_constants::c1_initstatesm, mo_mpr_global_variables::horizondepth_mhm, mo_global_variables::l1_aetcany, mo_global_variables::l1_aetsealed, mo_global_variables::l1_aetsoil, mo_global_variables::l1_baseflow, mo_global_variables::l1_fastrunoff, mo_global_variables::l1_infilsoil, mo_global_variables::l1_inter, mo_global_variables::l1_melt, mo_global_variables::l1_neutrons, mo_global_variables::l1_percol, mo_global_variables::l1_pet_calc, mo_global_variables::l1_preeffect, mo_global_variables::l1_rain, mo_global_variables::l1_runoffseal, mo_global_variables::l1_satstw, mo_global_variables::l1_sealstw, mo_global_variables::l1_slowrunoff, mo_global_variables::l1_snow, mo_global_variables::l1_snowpack, mo_global_variables::l1_soilmoist, mo_global_variables::l1_throughfall, mo_global_variables::l1_total_runoff, mo_global_variables::l1_unsatstw, mo_mpr_global_variables::nsoilhorizons_mhm, mo_common_constants::p1_initstatefluxes, mo_mpr_constants::p2_initstatefluxes, mo_mpr_constants::p3_initstatefluxes, mo_mpr_constants::p4_initstatefluxes, and mo_mpr_constants::p5_initstatefluxes.

Referenced by `mo_startup::mhm_initialize()`.

Here is the caller graph for this function:



15.24.2.2 variables_default_init()

```
subroutine, public mo_init_states::variables_default_init ( )
```

Default initialization mHM related L1 variables.

Default initialization of mHM related L1 variables (e.g., states, fluxes, and parameters) as per given constant values given in [mo_mhm_constants](#). Variables initialized here is defined in the [mo_global_variables.f90](#) file. Only Variables

that are defined in the `variables_alloc` subroutine are initialized 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_ncdf_restart`.

Authors

R. Kumar & J. Mai

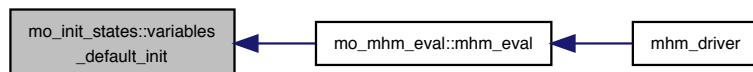
Date

Sep 2013

References `mo_mpr_constants::c1_initstatesm`, `mo_mpr_global_variables::horizondepth_mhm`, `mo_mpr_global_variables::l1_aeroresist`, `mo_global_variables::l1_aetcanopy`, `mo_global_variables::l1_aetsealed`, `mo_global_variables::l1_aetsoil`, `mo_mpr_global_variables::l1_alpha`, `mo_global_variables::l1_baseflow`, `mo_mpr_global_variables::l1_degday`, `mo_mpr_global_variables::l1_degdayinc`, `mo_mpr_global_variables::l1_degdaymax`, `mo_mpr_global_variables::l1_degdaynopre`, `mo_mpr_global_variables::l1_fasp`, `mo_global_variables::l1_fastrunoff`, `mo_mpr_global_variables::l1_froots`, `mo_mpr_global_variables::l1_fsealed`, `mo_mpr_global_variables::l1_harsamcoeff`, `mo_global_variables::l1_infilsoil`, `mo_global_variables::l1_inter`, `mo_mpr_global_variables::l1_jarvis_thresh_c1`, `mo_mpr_global_variables::l1_karstloss`, `mo_mpr_global_variables::l1_kbaseflow`, `mo_mpr_global_variables::l1_kfastflow`, `mo_mpr_global_variables::l1_kperco`, `mo_mpr_global_variables::l1_kslowflow`, `mo_mpr_global_variables::l1_maxinter`, `mo_global_variables::l1_melt`, `mo_global_variables::l1_neutrons`, `mo_global_variables::l1_percol`, `mo_global_variables::l1_pet_calc`, `mo_mpr_global_variables::l1_petlaicfactor`, `mo_global_variables::l1_preeffect`, `mo_mpr_global_variables::l1_prietaryalpha`, `mo_global_variables::l1_rain`, `mo_global_variables::l1_runoffseal`, `mo_global_variables::l1_satstw`, `mo_mpr_global_variables::l1_sealedthresh`, `mo_global_variables::l1_sealstw`, `mo_global_variables::l1_slowrunoff`, `mo_global_variables::l1_snow`, `mo_global_variables::l1_snowpack`, `mo_global_variables::l1_soilmoist`, `mo_mpr_global_variables::l1_soilmoistexp`, `mo_mpr_global_variables::l1_soilmoistfc`, `mo_mpr_global_variables::l1_soilmoistsat`, `mo_mpr_global_variables::l1_surfresist`, `mo_mpr_global_variables::l1_tempthresh`, `mo_global_variables::l1_throughfall`, `mo_global_variables::l1_total_runoff`, `mo_global_variables::l1_unsatstw`, `mo_mpr_global_variables::l1_unsatthresh`, `mo_mpr_global_variables::l1_wiltingpoint`, `mo_mpr_global_variables::nsoilhorizons_mhm`, `mo_common_constants::p1_initstatefluxes`, `mo_mpr_constants::p2_initstatefluxes`, `mo_mpr_constants::p3_initstatefluxes`, `mo_mpr_constants::p4_initstatefluxes`, and `mo_mpr_constants::p5_initstatefluxes`.

Referenced by `mo_mhm_eval::mhm_eval()`.

Here is the caller graph for this function:



15.25 mo_julian Module Reference

Julian date conversion routines.

Data Types

- interface `setcalendar`

Functions/Subroutines

- subroutine `setcalendarstring` (selector)

Set module private variable calendar.
- subroutine `setcalendarinteger` (selector)

Set module private variable calendar.
- pure integer(i4) function `selectcalendar` (selector)

Select a calendar.
- elemental subroutine, public `caldat` (julian, dd, mm, yy, `calendar`)

Day, month and year from Julian day in the current or given calendar.
- elemental subroutine, public `dec2date` (julian, dd, mm, yy, hh, nn, ss, `calendar`)

Day, month, year, hour, minute, and second from fractional Julian day in the current or given calendar.
- elemental real(dp) function, public `date2dec` (dd, mm, yy, hh, nn, ss, `calendar`)

Fractional Julian day from day, month, year, hour, minute, second in the current calendar.
- elemental integer(i4) function, public `julday` (dd, mm, yy, `calendar`)

Julian day from day, month and year in the current or given calendar.
- elemental subroutine, public `caldatjulian` (julian, dd, mm, yy)

Day, month and year from Julian day.
- elemental real(dp) function `date2decjulian` (dd, mm, yy, hh, nn, ss)

Fractional Julian day from day, month, year, hour, minute, second.
- elemental subroutine `dec2datejulian` (julian, dd, mm, yy, hh, nn, ss)

Day, month, year, hour, minute, and second from fractional Julian day.
- elemental integer(i4) function `juldayjulian` (dd, mm, yy)

Julian day from day, month and year.
- elemental integer(i4) function, public `ndays` (dd, mm, yy)

IMSL Julian day from day, month and year.
- elemental subroutine, public `ndyin` (julian, dd, mm, yy)

Day, month and year from IMSL Julian day.
- elemental subroutine `caldat360` (julian, dd, mm, yy)

Day, month and year from Julian day in a 360 day calendar.
- elemental integer(i4) function `julday360` (dd, mm, yy)

Julian day from day, month and year in a 360_day calendar.
- elemental subroutine `dec2date360` (julian, dd, mm, yy, hh, nn, ss)

Day, month, year, hour, minute, and second from fractional Julian day in a 360_day calendar.
- elemental real(dp) function `date2dec360` (dd, mm, yy, hh, nn, ss)

Fractional Julian day from day, month, year, hour, minute, second in 360 day calendar.
- elemental subroutine `caldat365` (julian, dd, mm, yy)

Day, month and year from Julian day in a 365 day calendar.
- elemental integer(i4) function `julday365` (dd, mm, yy)

Julian day from day, month and year in a 365_day calendar.
- elemental subroutine `dec2date365` (julian, dd, mm, yy, hh, nn, ss)

Day, month, year, hour, minute, and second from fractional Julian day in a 365_day calendar.
- elemental real(dp) function `date2dec365` (dd, mm, yy, hh, nn, ss)

Fractional Julian day from day, month, year, hour, minute, second in 365 day calendar.

Variables

- integer(i4), save, private `calendar` = 1

15.25.1 Detailed Description

Julian date conversion routines.

Julian date to and from day, month, year, and also from day, month, year, hour, minute, and second. Also convenience 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

15.25.2 Function/Subroutine Documentation

15.25.2.1 caldat()

```
elemental subroutine, public mo_julian::caldat (
    integer(i4), intent(in) julian,
    integer(i4), intent(out) dd,
    integer(i4), intent(out) mm,
    integer(i4), intent(out) yy,
    integer(i4), intent(in), optional calendar )
```

Day, month and year from Julian day in the current or given calendar.

Wrapper around the calendar specific caldat procedures. Inverse of the function julday. Here julian is input as a Julian Day Number, and the routine outputs dd, 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	<i>integer(i4) :: julday</i>	Julian day
out	<i>integer(i4) :: dd</i>	Day in month of Julian day
out	<i>integer(i4) :: mm</i>	Month in year of Julian day
out	<i>integer(i4) :: yy</i>	Year of Julian day
in	<i>integer(i4) :: calendar</i>	The calendar to use, the global calendar will be used by default

Author

Written, David Schaefer

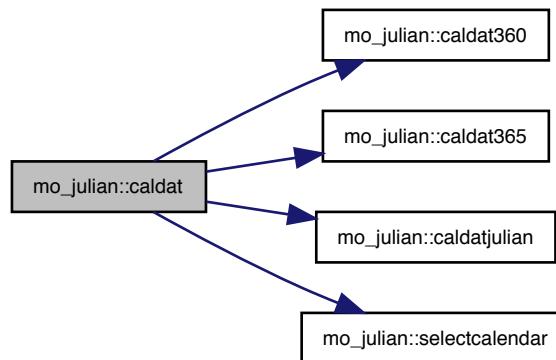
Date

Jan 2015

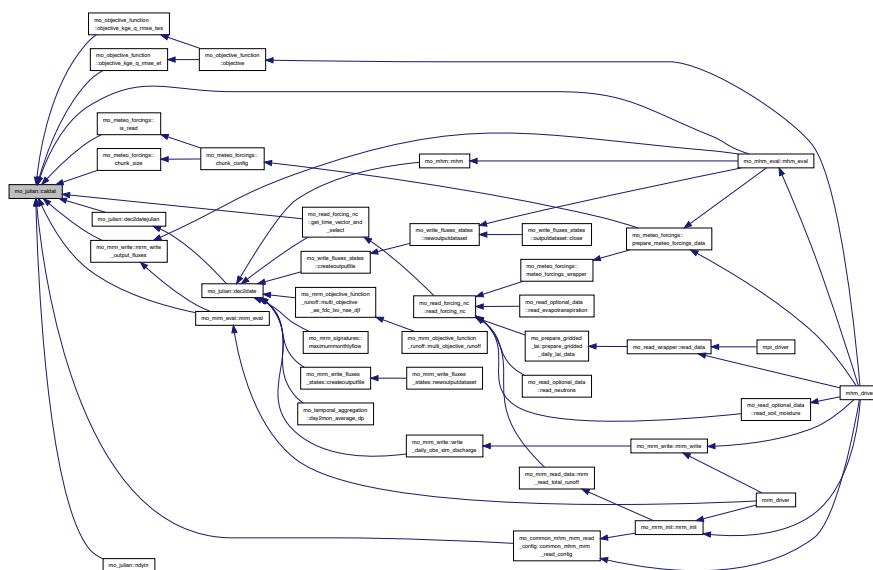
References `caldat360()`, `caldat365()`, `caldatjulian()`, and `selectcalendar()`.

Referenced by `mo_meteo_forcings::chunk_size()`, `mo_common_mhm_mrm_read_config::common_mhm_mrm_read_config()`, `dec2datejulian()`, `mo_read_forcing_nc::get_time_vector_and_select()`, `mo_meteo_forcings::is_read()`, `mo_mhm_eval::mhm_eval()`, `mo_mrm_eval::mrm_eval()`, `mo_mrm_write::mrm_write_output_fluxes()`, `ndyin()`, `mo_objective_function::objective_kge_q_rmse_et()`, and `mo_objective_function::objective_kge_q_rmse_tws()`.

Here is the call graph for this function:



Here is the caller graph for this function:



15.25.2.2 caldat360()

```
elemental subroutine mo_julian::caldat360 (
    integer(i4), intent(in) julian,
    integer(i4), intent(out) dd,
    integer(i4), intent(out) mm,
    integer(i4), intent(out) yy ) [private]
```

Day, month and year from Julian day in a 360 day calendar.

Inverse of the function julday360. Here julian is input as a Julian Day Number, and the routine outputs dd, mm, and yy as the day, month, and year on which the specified Julian Day started at noon. The zeroth Julian Day here is 01.01.0000

Parameters

in	<i>integer(i4) :: julday</i>	Julian day
out	<i>integer(i4) :: dd</i>	Day in month of Julian day
out	<i>integer(i4) :: mm</i>	Month in year of Julian day
out	<i>integer(i4) :: yy</i>	Year of Julian day

Author

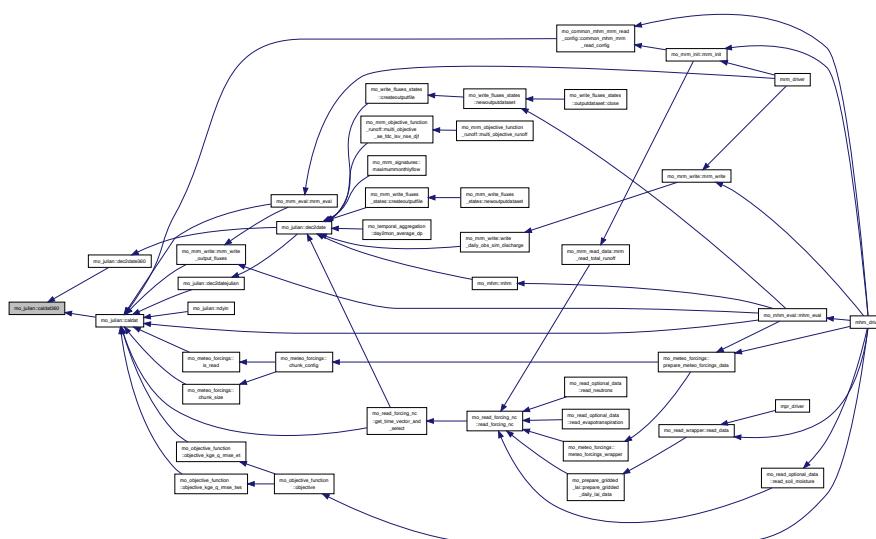
Written, David Schaefer

Date

Oct 2015

Referenced by caldat(), and dec2date360().

Here is the caller graph for this function:



15.25.2.3 caldat365()

```
elemental subroutine mo_julian::caldat365 (  
    integer(i4), intent(in) julian,  
    integer(i4), intent(out) dd,  
    integer(i4), intent(out) mm,  
    integer(i4), intent(out) yy ) [private]
```

Day, month and year from Julian day in a 365 day calendar.

Inverse of the function `julday365`. Here `julian` is input as a Julian Day Number, and the routine outputs `dd`, `mm`, and `yy` as the day, month, and year on which the specified Julian Day started at noon. The zeroth Julian Day here is 01.01.0000

Parameters

in	<i>integer(i4) :: julday</i>	Julian day
out	<i>integer(i4) :: dd</i>	Day in month of Julian day
out	<i>integer(i4) :: mm</i>	Month in year of Julian day
out	<i>integer(i4) :: yy</i>	Year of Julian day

Author

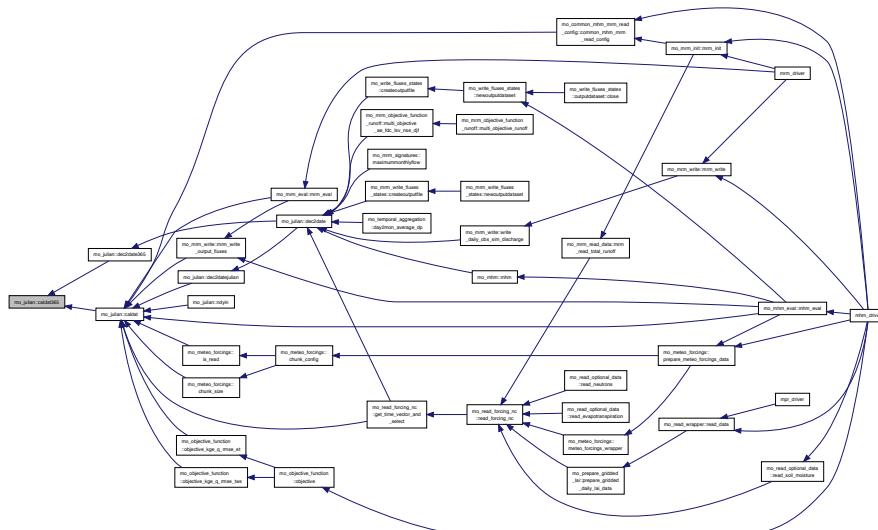
Written, David Schaefer

Date

Dec 2015

Referenced by caldat(), and dec2date365().

Here is the caller graph for this function:



15.25.2.4 caldatjulian()

```
elemental subroutine, public mo_julian::caldatjulian (
```

```
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 dd, 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	<i>integer(i4) :: Julday</i>	Julian day
out	<i>integer(i4) :: dd</i>	Day in month of Julian day
out	<i>integer(i4) :: mm</i>	Month in year of Julian day
out	<i>integer(i4) :: yy</i>	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 caldatJulian together for integer Julian days.

Author

Written, Matthias Cuntz - modified julday from Numerical Recipes

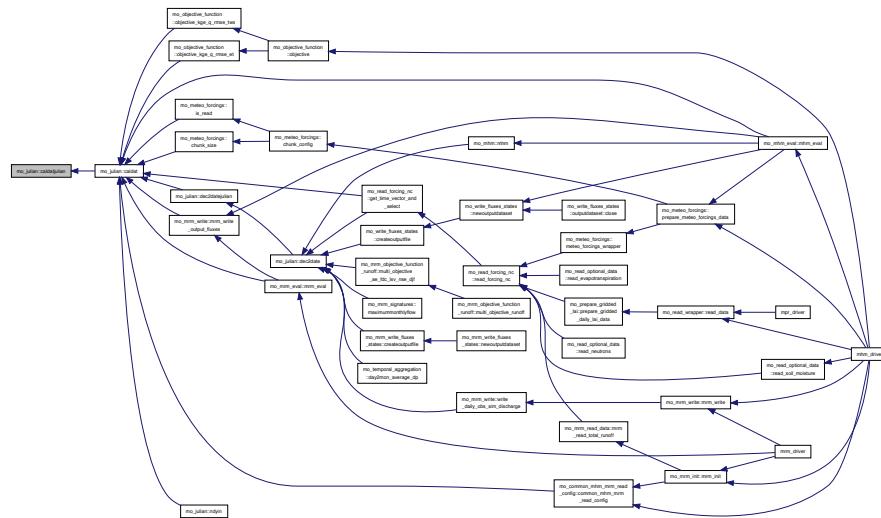
Date

Dec 2011 Modified Matthias Cuntz, May 2014 - changed to new algorithm with astronomical units removed
numerical recipes David Schaefer, Jan 2016 - renamed procedure

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

Referenced by caldat().

Here is the caller graph for this function:



15.25.2.5 date2dec()

```
elemental real(dp) function, public mo Julian::date2dec (
```

integer(i4), intent(in), optional dd,
integer(i4), intent(in), optional mm,
integer(i4), intent(in), optional yy,
integer(i4), intent(in), optional hh,
integer(i4), intent(in), optional nn,
integer(i4), intent(in), optional ss,
integer(i4), intent(in), optional calendar)

Fractional Julian day from day, month, year, hour, minute, second in the current calendar.

Wrapper around the calendar specific date2dec procedures. In this routine date2dec returns the fractional Julian Day that begins at noon of the calendar date specified by month mm, day dd, and year yy, all integer variables. The zeroth Julian Day depends on the called procedure. See their documentation for details.

Parameters

in	<i>integer(i4), optional :: dd</i>	Day in month of Julian day (default: 1)
in	<i>integer(i4), optional :: mm</i>	Month in year of Julian day (default: 1)
in	<i>integer(i4), optional :: yy</i>	Year of Julian day (default: 1)
in	<i>integer(i4), optional :: hh</i>	Hours of Julian day (default: 0)
in	<i>integer(i4), optional :: nn</i>	Minutes of hour of Julian day (default: 0)
in	<i>integer(i4), optional :: ss</i>	Secondes of minute of hour of Julian day (default: 0)
in	<i>integer(i4), optional :: calendar</i>	The calendar to use, the global calendar will be used by default

Returns

```
real(dp) :: date2dec ! Fractional Julian day
```

Author

Written, David Schaefer

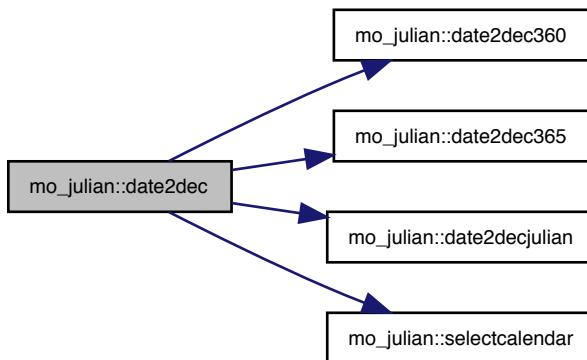
Date

Jan 2015

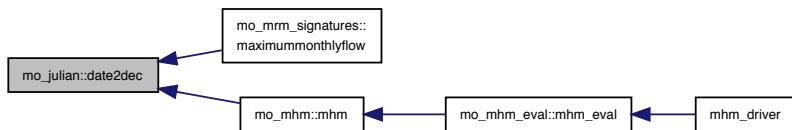
References `date2dec360()`, `date2dec365()`, `date2decjulian()`, and `selectcalendar()`.

Referenced by `mo_mrm_signatures::maximummonthlyflow()`, and `mo_mhm::mhm()`.

Here is the call graph for this function:



Here is the caller graph for this function:



15.25.2.6 date2dec360()

```
elemental real(dp) function mo_julian::date2dec360 (
    integer(i4), intent(in), optional dd,
    integer(i4), intent(in), optional mm,
```

```
integer(i4), intent(in), optional yy,
integer(i4), intent(in), optional hh,
integer(i4), intent(in), optional nn,
integer(i4), intent(in), optional ss ) [private]
```

Fractional Julian day from day, month, year, hour, minute, second in 360 day calendar.

In this routine date2dec360 returns the fractional Julian Day that begins at noon of the calendar date specified by month mm, day dd, and year yy, all integer variables. The zeroth Julian Day is 01.01.0000 at noon.

Parameters

in	<i>integer(i4), optional :: dd</i>	Day in month of Julian day (default: 1)
in	<i>integer(i4), optional :: mm</i>	Month in year of Julian day (default: 1)
in	<i>integer(i4), optional :: yy</i>	Year of Julian day (default: 1)
in	<i>integer(i4), optional :: hh</i>	Hours of Julian day (default: 0)
in	<i>integer(i4), optional :: nn</i>	Minutes of hour of Julian day (default: 0)
in	<i>integer(i4), optional :: ss</i>	Seconds of minute of hour of Julian day (default: 0)

Returns

real(dp) :: date2dec360 ! Fractional Julian day

Author

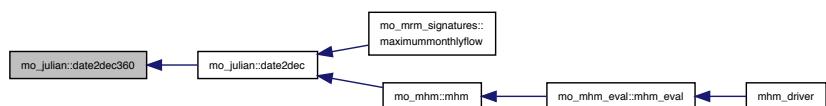
Written, David Schaefer

Date

Oct 2015

Referenced by date2dec().

Here is the caller graph for this function:



15.25.2.7 date2dec365()

```
elemental real(dp) function mo_julian::date2dec365 (
    integer(i4), intent(in), optional dd,
    integer(i4), intent(in), optional mm,
    integer(i4), intent(in), optional yy,
    integer(i4), intent(in), optional hh,
    integer(i4), intent(in), optional nn,
    integer(i4), intent(in), optional ss ) [private]
```

Fractional Julian day from day, month, year, hour, minute, second in 365 day calendar.

In this routine date2dec365 returns the fractional Julian Day that begins at noon of the calendar date specified by month mm, day dd, and year yy, all integer variables. The zeroth Julian Day is 01.01.0000 at noon.

Parameters

in	<i>integer(i4), optional :: dd</i>	Day in month of Julian day (default: 1)
in	<i>integer(i4), optional :: mm</i>	Month in year of Julian day (default: 1)
in	<i>integer(i4), optional :: yy</i>	Year of Julian day (default: 1)
in	<i>integer(i4), optional :: hh</i>	Hours of Julian day (default: 0)
in	<i>integer(i4), optional :: nn</i>	Minutes of hour of Julian day (default: 0)
in	<i>integer(i4), optional :: ss</i>	Seconds of minute of hour of Julian day (default: 0)

Returns

real(dp) :: date2dec365 ! Fractional Julian day

Author

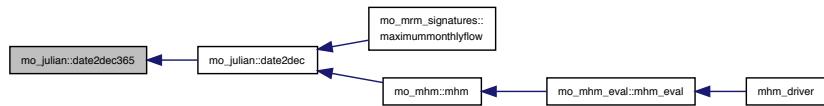
Written, David Schaefer

Date

Dec 2015

Referenced by *date2dec()*.

Here is the caller graph for this function:



15.25.2.8 date2decjulian()

```
elemental real(dp) function mo_julian::date2decjulian (
    integer(i4), intent(in), optional dd,
    integer(i4), intent(in), optional mm,
    integer(i4), intent(in), optional yy,
    integer(i4), intent(in), optional hh,
    integer(i4), intent(in), optional nn,
    integer(i4), intent(in), optional ss ) [private]
```

Fractional Julian day from day, month, year, hour, minute, second.

In this routine *date2decJulian* returns the fractional Julian Day that begins at noon of the calendar date specified by month mm, day dd, and year yy, all integer variables. The zeroth Julian Day is 01.01.-4712 at noon, i.e. the 1st January 4713 BC 12:00:00 h. Julian day definition starts at noon of the 1st January 4713 BC.

Here, the astronomical definition is used, i.e. the year 1 BC (historic) is counted as 0 (astronomic), 2 BC is -1, etc. This means that Julian day definition starts as 01.01.-4712 in astronomical units.

Parameters

in	<i>integer(i4), optional :: dd</i>	Day in month of Julian day (default: 1)
in	<i>integer(i4), optional :: mm</i>	Month in year of Julian day (default: 1)
in	<i>integer(i4), optional :: yy</i>	Year of Julian day (default: 1)
in	<i>integer(i4), optional :: hh</i>	Hours of Julian day (default: 0)
in	<i>integer(i4), optional :: nn</i>	Minutes of hour of Julian day (default: 0)
in	<i>integer(i4), optional :: ss</i>	Seconds 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 *caldatJulian* together for integer Julian days.

Author

Written, Matthias Cuntz

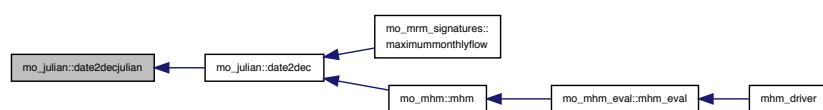
Date

Jan 2013 Modified Matthias Cuntz, May 2014 - changed to new algorithm with astronomical units removed
numerical recipes David Schaefer, Jan 2016 - renamed procedure

References *mo_kind::i8*.

Referenced by *date2dec()*.

Here is the caller graph for this function:



15.25.2.9 dec2date()

```
elemental subroutine, public mo Julian::dec2date (
    real(dp), intent(in) julian,
```

```
integer(i4), intent(out), optional dd,
integer(i4), intent(out), optional mm,
integer(i4), intent(out), optional yy,
integer(i4), intent(out), optional hh,
integer(i4), intent(out), optional nn,
integer(i4), intent(out), optional ss,
integer(i4), intent(in), optional calendar )
```

Day, month, year, hour, minute, and second from fractional Julian day in the current or given calendar.

Wrapper around the calendar specific dec2date procedures. Inverse of the function date2dec. Here dec2date is input as a fractional Julian Day. The routine outputs dd, mm, yy, hh, nn, ss as the day, month, year, hour, minute, and second on which the specified Julian Day started at noon. The zeroth Julian Day depends on the called procedure. See their documentation for details.

Parameters

in	<i>real(dp) :: fJulian</i>	fractional Julian day
in	<i>integer(i4) :: calendar</i>	The calendar to use, the global calendar will be used by default
out	<i>integer(i4), optional :: dd</i>	Day in month of Julian day
out	<i>integer(i4), optional :: mm</i>	Month in year of Julian day
out	<i>integer(i4), optional :: yy</i>	Year of Julian day
out	<i>integer(i4), optional :: hh</i>	Hour of Julian day
out	<i>integer(i4), optional :: nn</i>	Minute in hour of Julian day
out	<i>integer(i4), optional :: ss</i>	Second in minute of hour of Julian day

Author

Written, David Schaefer

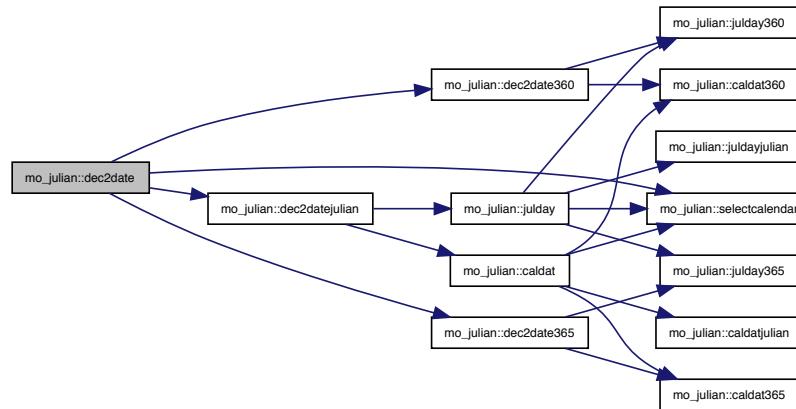
Date

Jan 2015

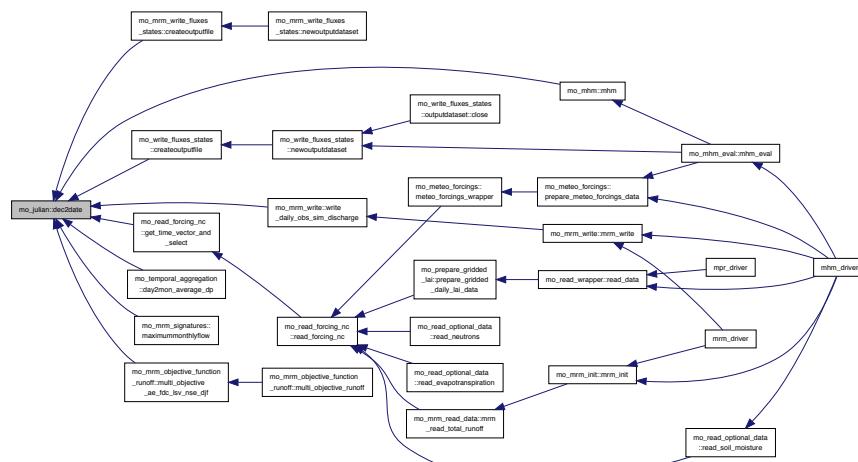
References dec2date360(), dec2date365(), dec2datejulian(), and selectcalendar().

Referenced by mo_mrm_write_fluxes_states::createoutputfile(), mo_write_fluxes_states::createoutputfile(), mo_temporal_aggregation::day2mon_average_dp(), mo_read_forcing_nc::get_time_vector_and_select(), mo_mrm_signatures::maximummonthlyflow(), mo_mhm::mhm(), mo_mrm_objective_function_runoff::multi_objective_ae_fdc_lsv_nse_djf(), and mo_mrm_write::write_daily_obs_sim_discharge().

Here is the call graph for this function:



Here is the caller graph for this function:



15.25.2.10 dec2date360()

```

elemental subroutine mo Julian::dec2date360 (
    real(dp), intent(in) julian,
    integer(i4), intent(out), optional dd,
    integer(i4), intent(out), optional mm,
    integer(i4), intent(out), optional yy,
    integer(i4), intent(out), optional hh,
    integer(i4), intent(out), optional nn,
    integer(i4), intent(out), optional ss ) [private]
  
```

Day, month, year, hour, minute, and second from fractional Julian day in a 360_day calendar.

Inverse of the function date2dec360. Here dec2date360 is input as a fractional Julian Day. The routine outputs dd,

mm, yy, hh, nn, ss as the day, month, year, hour, minute, and second on which the specified Julian Day started at noon. The zeroth Julian Day is 01.01.0000 at noon.

Parameters

in	<i>real(dp) :: fJulian</i>	fractional Julian day
out	<i>integer(i4), optional :: dd</i>	Day in month of Julian day
out	<i>integer(i4), optional :: mm</i>	Month in year of Julian day
out	<i>integer(i4), optional :: yy</i>	Year of Julian day
out	<i>integer(i4), optional :: hh</i>	Hour of Julian day
out	<i>integer(i4), optional :: nn</i>	Minute in hour of Julian day
out	<i>integer(i4), optional :: ss</i>	Second in minute of hour of Julian day

Author

Written, David Schaefer

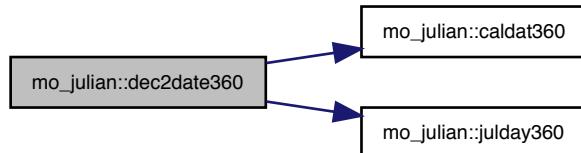
Date

Oct 2015

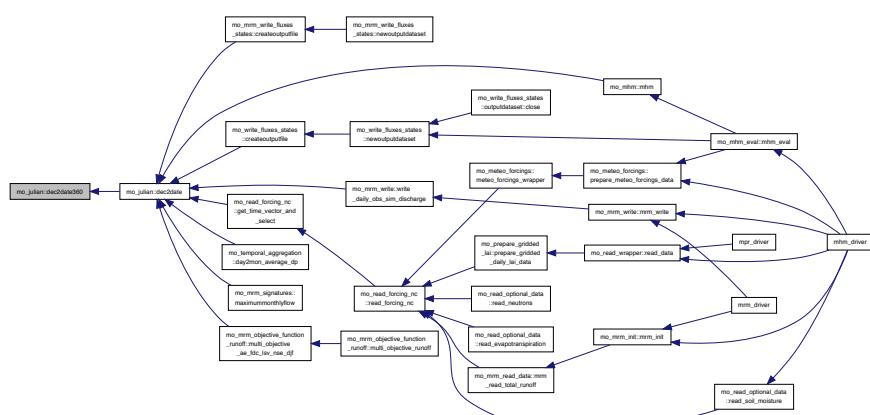
References `caldat360()`, `mo` `kind::i4`, and `julday360()`.

Referenced by dec2date().

Here is the call graph for this function:



Here is the caller graph for this function:



15.25.2.11 dec2date365()

```
elemental subroutine mo_julian::dec2date365 (
    real(dp), intent(in) julian,
    integer(i4), intent(out), optional dd,
    integer(i4), intent(out), optional mm,
    integer(i4), intent(out), optional yy,
    integer(i4), intent(out), optional hh,
    integer(i4), intent(out), optional nn,
    integer(i4), intent(out), optional ss ) [private]
```

Day, month, year, hour, minute, and second from fractional Julian day in a 365_day calendar.

Inverse of the function date2dec. Here dec2date365 is input as a fractional Julian Day. The routine outputs dd, mm, yy, hh, nn, ss as the day, month, year, hour, minute, and second on which the specified Julian Day started at noon. The zeroth Julian Day is 01.01.0000 at noon.

Parameters

in	<i>real(dp) :: fJulian</i>	fractional Julian day
out	<i>integer(i4), optional :: dd</i>	Day in month of Julian day
out	<i>integer(i4), optional :: mm</i>	Month in year of Julian day
out	<i>integer(i4), optional :: yy</i>	Year of Julian day
out	<i>integer(i4), optional :: hh</i>	Hour of Julian day
out	<i>integer(i4), optional :: nn</i>	Minute in hour of Julian day
out	<i>integer(i4), optional :: ss</i>	Second in minute of hour of Julian day

Author

Written, David Schaefer

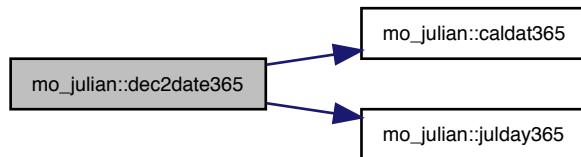
Date

Dec 2015

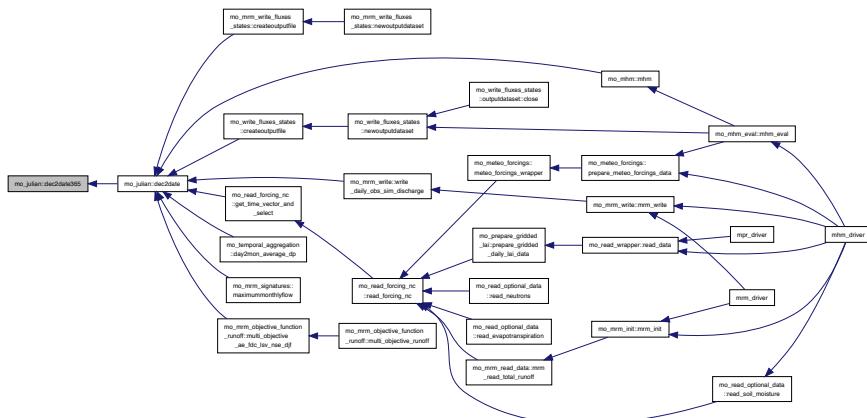
References caldat365(), mo_kind::i4, and julday365().

Referenced by dec2date().

Here is the call graph for this function:



Here is the caller graph for this function:



15.25.2.12 dec2datejulian()

```
elemental subroutine mo Julian :: dec2dateJulian (  
    real(dp), intent(in) Julian,  
    integer(i4), intent(out), optional dd,  
    integer(i4), intent(out), optional mm,  
    integer(i4), intent(out), optional yy,  
    integer(i4), intent(out), optional hh,  
    integer(i4), intent(out), optional nn,  
    integer(i4), intent(out), optional ss ) [private]
```

Day, month, year, hour, minute, and second from fractional Julian day.

Inverse of the function date2decJulian. Here dec2dateJulian is input as a fractional Julian Day, which starts at noon of the 1st January 4713 BC, i.e. 01.01.-4712. The routine outputs dd, mm, yy, hh, nn, ss as the day, month, year, hour, minute, and second on which the specified Julian Day started at noon. The zeroth Julian Day is 01.01.-4712 at noon, i.e. the 1st January 4713 BC at noon. Julian day definition starts at 1st January 4713 BC. Here, the astronomical definition is used, i.e. the year 1 BC (historic) is counted as 0 (astronomic), 2 BC is -1, etc. This means that Julian day definition starts as 01.01.-4712 in astronomical units.

Parameters

in	<i>real(dp) :: fJulian</i>	fractional Julian day
out	<i>integer(i4), optional :: dd</i>	Day in month of Julian day
out	<i>integer(i4), optional :: mm</i>	Month in year of Julian day
out	<i>integer(i4), optional :: yy</i>	Year of Julian day
out	<i>integer(i4), optional :: hh</i>	Hour of Julian day
out	<i>integer(i4), optional :: nn</i>	Minute in hour of Julian day
out	<i>integer(i4), optional :: ss</i>	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, 3 BC is -2, and so on.

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 caldatJulian together for integer Julian days.

Author

Written, Matthias Cuntz

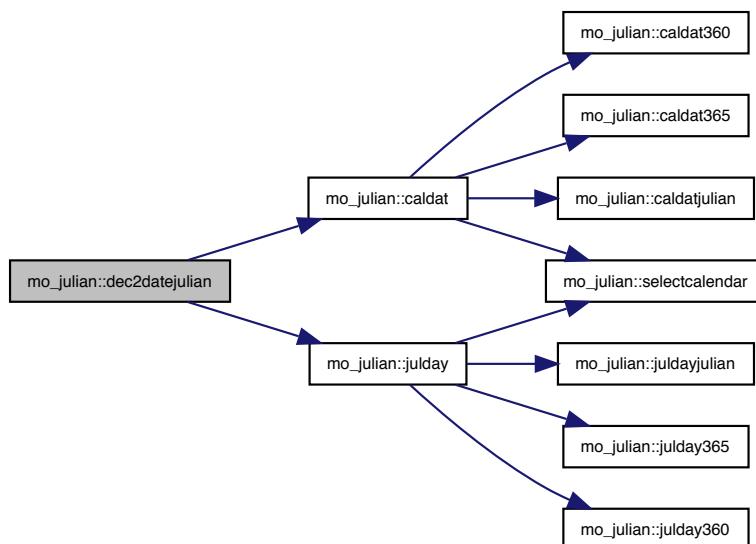
Date

Jan 2013 Modified Matthias Cuntz, May 2014 - changed to new algorithm with astronomical units removed
numerical recipes David Schaefer, Jan 2016 - renamed procedure

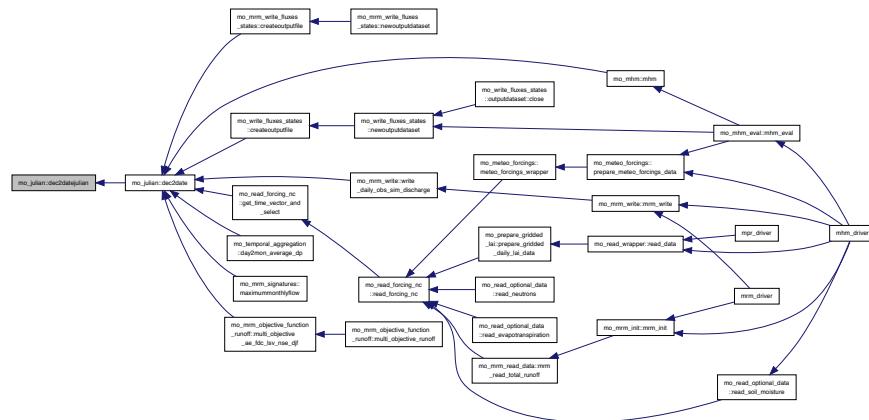
References caldat(), mo_kind::i4, mo_kind::i8, and julday().

Referenced by dec2date().

Here is the call graph for this function:



Here is the caller graph for this function:



15.25.2.13 julday()

```
elemental integer(i4) function, public mo_julian::julday (
```

integer(i4), intent(in) dd,
integer(i4), intent(in) mm,
integer(i4), intent(in) yy,
integer(i4), intent(in), optional calendar)

Julian day from day, month and year in the current or given calendar.

Wrapper around the calendar specific julday procedures. In this routine julday returns the Julian Day Number that begins at noon of the calendar date specified by month mm, day dd, and year yy, all integer variables. The zeroth Julian Day depends on the called procedure. See their documentation for details.

Parameters

in	<i>integer(i4) :: dd</i>	Day in month of Julian day
in	<i>integer(i4) :: mm</i>	Month in year of Julian day
in	<i>integer(i4) :: yy</i>	Year of Julian day
in	<i>integer(i4), optional :: calendar</i>	The calendar to use, the global calendar will be used by default

Returns

```
integer(i4) :: julian ! Julian day
```

Author

Written, David Schaefer

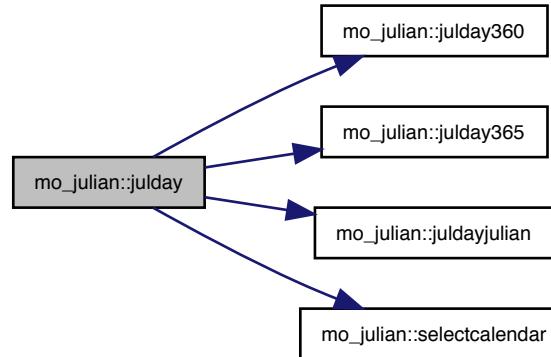
Date

Jan 2015

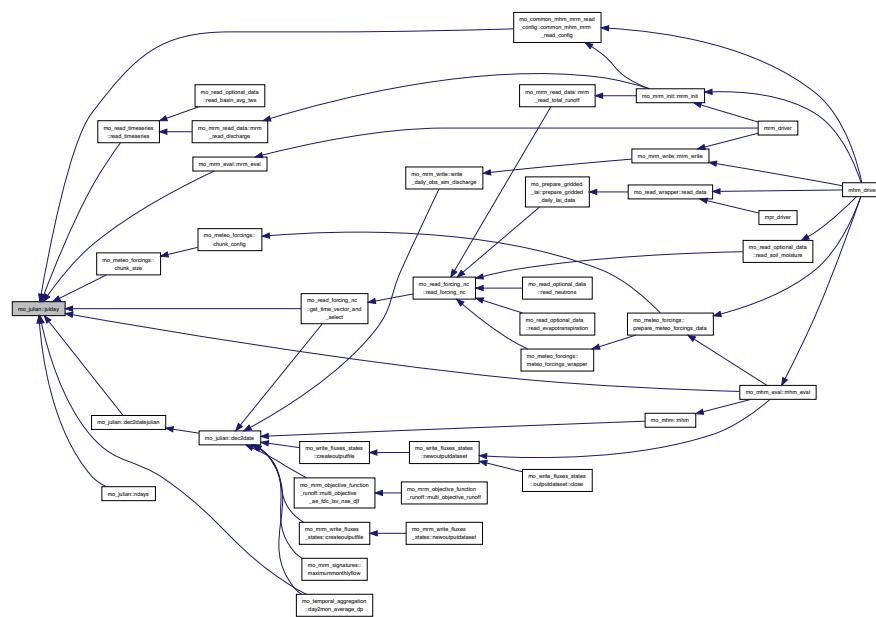
References `julday360()`, `julday365()`, `juldayjulian()`, and `selectcalendar()`.

Referenced by `mo_meteo_forcings::chunk_size()`, `mo_common_mhm_mrm_read_config::common_mhm_mrm::read_config()`, `mo_temporal_aggregation::day2mon_average_dp()`, `dec2datejulian()`, `mo_read_forcing_nc::get_time_vector_and_select()`, `mo_mhm_eval::mhm_eval()`, `mo_mrm_eval::mrm_eval()`, `ndays()`, and `mo_read::timeseries::read_timeseries()`.

Here is the call graph for this function:



Here is the caller graph for this function:



15.25.2.14 julday360()

```

elemental integer(i4) function mo Julian julday360 (
    integer(i4), intent(in) dd,
  
```

```
integer(i4), intent(in) mm,  
integer(i4), intent(in) yy ) [private]
```

Julian day from day, month and year in a 360-day calendar.

In this routine `Julday360` returns the Julian Day Number that begins at noon of the calendar date specified by month mm, day dd, and year yy, all integer variables. The zeroth Julian Day is 01.01.0000

Parameters

in	<i>integer(i4) :: dd</i>	Day in month of Julian day
in	<i>integer(i4) :: mm</i>	Month in year of Julian day
in	<i>integer(i4) :: yy</i>	Year of Julian day

Returns

```
integer(i4) :: julian ! Julian day
```

Author

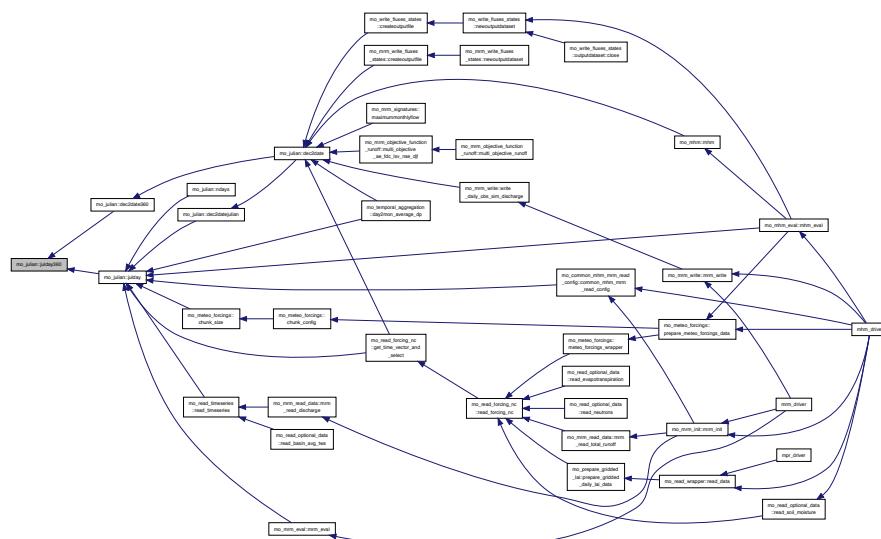
Written, David Schaefer

Date

Oct 2015

Referenced by `dec2date360()`, and `julday()`.

Here is the caller graph for this function:



15.25.2.15 `julday365()`

```
elemental integer(i4) function mo Julian::julday365 (
```

```
integer(i4), intent(in) mm,
integer(i4), intent(in) yy ) [private]
```

Julian day from day, month and year in a 365_day calendar.

In this routine julday365 returns the Julian Day Number that begins at noon of the calendar date specified by month mm, day dd, and year yy, all integer variables. The zeroth Julian Day is 01.01.0000

Parameters

in	integer(i4) :: dd	Day in month of Julian day
in	integer(i4) :: mm	Month in year of Julian day
in	integer(i4) :: yy	Year of Julian day

Returns

```
integer(i4) :: julian ! Julian day
```

Author

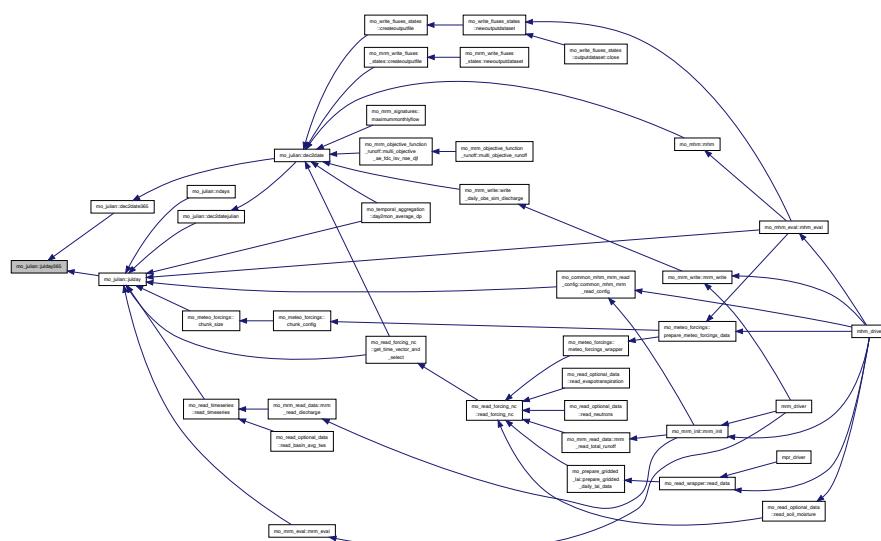
Written, David Schaefer

Date

Dec 2015

Referenced by dec2date365(), and julday().

Here is the caller graph for this function:



15.25.2.16 juldayjulian()

```
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	<i>integer(i4) :: dd</i>	Day in month of Julian day
in	<i>integer(i4) :: mm</i>	Month in year of Julian day
in	<i>integer(i4) :: yy</i>	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 caldatJulian together for integer Julian days.

Author

Written, Matthias Cuntz - modified julday from Numerical Recipes

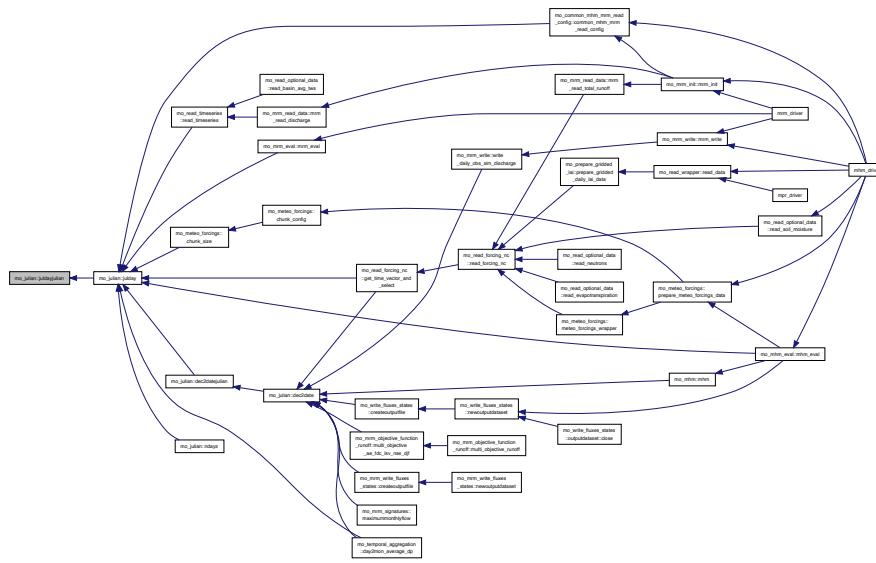
Date

Dec 2011 Modified Matthias Cuntz, May 2014 - changed to new algorithm with astronomical units removed
numerical recipes David Schaefer, Jan 2016 - renamed procedure

References mo_kind::i8.

Referenced by `julday()`.

Here is the caller graph for this function:



15.25.2.17 `ndays()`

```
elemental integer(i4) function, public mo Julian:::ndays (
```

integer(i4), intent(in) dd,
integer(i4), intent(in) mm,
integer(i4), intent(in) yy)

IMSL Julian day from day, month and year.

In this routine `ndays` returns the IMSL Julian Day Number. Julian days begin at noon of the calendar date specified by month `mm`, day `dd`, and year `yy`, all integer variables. IMSL treats 01.01.1900 as a reference and assigns a Julian day 0 to it.
$$\text{ndays} = \text{julday}(\text{dd}, \text{mm}, \text{yy}) - \text{julday}(01, 01, 1900)$$

Parameters

in	<i>integer(i4) :: dd</i>	Day in month of IMSL Julian day
in	<i>integer(i4) :: mm</i>	Month in year of IMSL Julian day
in	<i>integer(i4) :: yy</i>	Year of IMSL Julian day

Returns

integer(i4) :: julian — IMSL Julian day, i.e. days before or after 01.01.1900

Author

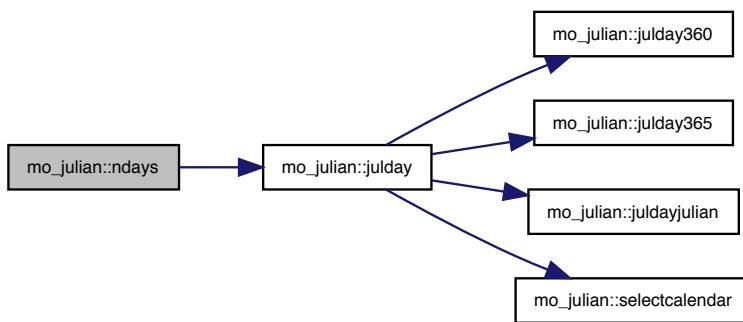
Written, Matthias Cuntz

Date

Dec 2011

References `julday()`.

Here is the call graph for this function:

15.25.2.18 `ndyin()`

```

elemental subroutine, public mo_julian::ndyin (
    integer(i4), intent(in) julian,
    integer(i4), intent(out) dd,
    integer(i4), intent(out) mm,
    integer(i4), intent(out) yy )
  
```

Day, month and year from IMSL Julian day.

Inverse of the function `ndys`. Here IMSL 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	<code>integer(i4) :: julian</code>	IMSL Julian day, i.e. days before or after 01.01.1900
out	<code>integer(i4) :: dd</code>	Day in month of IMSL Julian day
out	<code>integer(i4) :: mm</code>	Month in year of IMSL Julian day
out	<code>integer(i4) :: yy</code>	Year of IMSL Julian day

Author

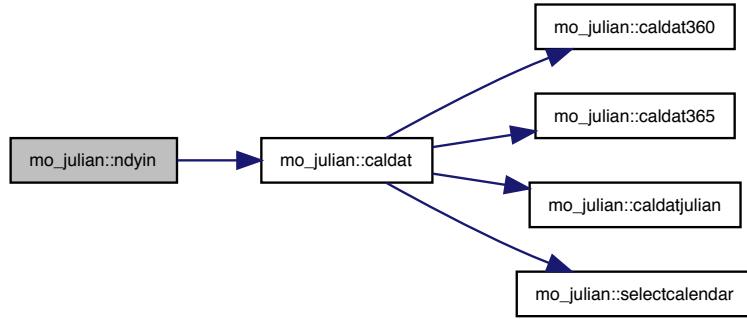
Written, Matthias Cuntz

Date

Dec 2011

References `caldat()`.

Here is the call graph for this function:

**15.25.2.19 `selectcalendar()`**

```
pure integer(i4) function mo_julian::selectcalendar (
    integer(i4), intent(in), optional selector ) [private]
```

Select a calendar.

Returns a valid calendar index, based on the given optional argument and/or the module global private variable `calendar`. If an invalid selector is passed, its value is ignored and the global calendar value retuned instead.

Parameters

in	<code>integer(i4), optional :: selector</code>	Calendar selector {1 2 3}
----	--	---------------------------

Author

Written, David Schaefer

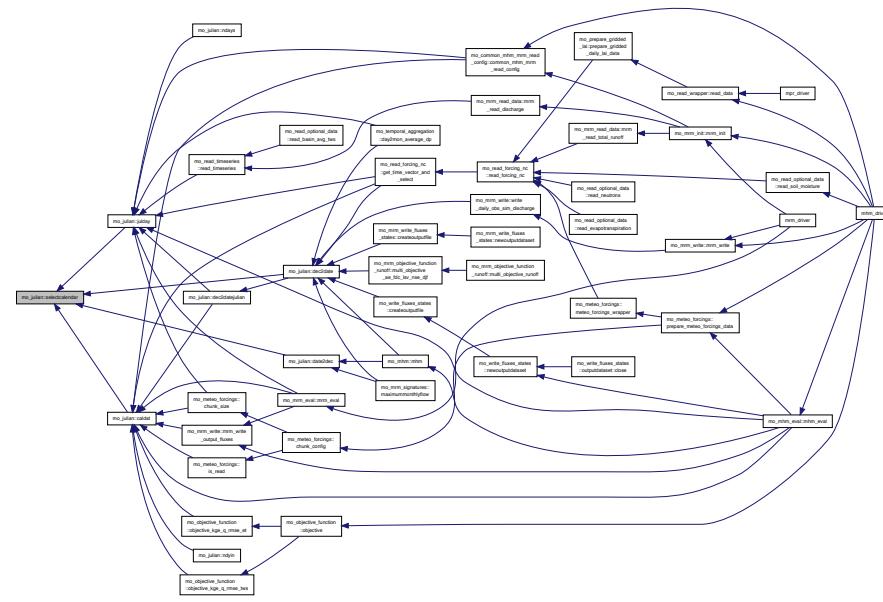
Date

Jan 2015

References calendar.

Referenced by caldat(), date2dec(), dec2date(), and julday().

Here is the caller graph for this function:



15.25.2.20 setcalendarinteger()

```
subroutine mo_julian::setcalendarinteger (
    integer(i4), intent(in) selector ) [private]
```

Set module private variable calendar.

Parameters

in	integer(i4) :: selector	{1 2 3}
----	-------------------------	---------

Author

Written, David Schaefer

Date

Jan 2015

References calendar.

Referenced by setcalendarstring().

Here is the caller graph for this function:



15.25.2.21 setcalendarstring()

```
subroutine mo_julian::setcalendarstring (
    character(*), intent(in) selector ) [private]
```

Set module private variable calendar.

Parameters

in	character(len=*) :: selector	{"julian" "365day" "360day"}
----	------------------------------	------------------------------

Author

Written, David Schaefer

Date

Jan 2015

References setcalendarinteger().

Here is the call graph for this function:



15.25.3 Variable Documentation

15.25.3.1 calendar

```
integer(i4), save, private mo_julian::calendar = 1 [private]
```

Referenced by selectcalendar(), and setcalendarinteger().

15.26 mo_kind Module Reference

Define number representations.

Data Types

- type `sprs2_dp`
Double Precision Numerical Recipes types for sparse arrays.
- type `sprs2_sp`
Single Precision Numerical Recipes types for sparse arrays.

Variables

- integer, parameter `i1` = SELECTED_INT_KIND(2)
1 Byte Integer Kind
- integer, parameter `i2` = c_short
2 Byte Integer Kind
- integer, parameter `i4` = c_int
4 Byte Integer Kind
- integer, parameter `i8` = c_long_long
8 Byte Integer Kind
- integer, parameter `sp` = c_float
Single Precision Real Kind.
- integer, parameter `dp` = c_double
Double Precision Real Kind.
- integer, parameter `spc` = c_float_complex
Single Precision Complex Kind.
- integer, parameter `dpc` = c_double_complex
Double Precision Complex Kind.
- integer, parameter `lgt` = KIND(.true.)
Logical Kind.

15.26.1 Detailed Description

Define number representations.

This module declares the desired ranges and precisions of the number representations, such as single precision or double precision, 32-bit or 46-bit integer, etc. It confirms mostly with the nrtype module of Numerical Recipes in f90.

Authors

Juliane Mai, Matthias Cuntz, Nov 2011

Date

2011-2014

Copyright

GNU Lesser General Public License <http://www.gnu.org/licenses/>

15.26.2 Variable Documentation

15.26.2.1 dp

```
integer, parameter mo_kind::dp = c_double
```

Double Precision Real Kind.

Referenced by mo_mrm_routing::add_inflow(), mo_sce::cce(), mo_sce::chkcst(), mo_sce::comp(), mo_anneal::dchange_dp(), mo_dds::dds(), mo_optimization_utils::eval_interface::eval_interface(), mo_read_forcing_nc::get_time_vector_and_select(), mo_sce::getpnt(), mo_anneal::gettemperature_dp(), mo_dds::mdds(), mhm_driver(), mrm_driver(), mo_mrm_eval::mrm_eval(), mo_mrm_init::mrm_init_param(), mo_mrm_restart::mrm_read_restart_config(), mo_mrm_init::mrm_update_param(), mo_mrm_write::mrm_write_output_fluxes(), mo_dds::neigh_value(), mo_optimization_utils::objective_interface::objective_interface(), mo_anneal::pargen_anneal_dp(), mo_mcmc::pargen_dp(), mo_mcmc::pargennorm_dp(), mo_sce::parstt(), mo_spatialsimilarity::pd_dp(), mo_read_forcing_nc::read_forcing_nc(), mo_common_restart::read_grid_info(), mo_restart::read_restart_states(), mo_soil_database::read_soil_lut(), mo_read_forcing_nc::read_weights_nc(), mo_sce::sce(), mo_sce::sort_matrix(), mo_mpr_restart::unpack_field_and_write_1d_dp(), mo_restart::unpack_field_and_write_1d_dp(), mo_mpr_restart::unpack_field_and_write_2d_dp(), mo_restart::unpack_field_and_write_2d_dp(), mo_mpr_restart::unpack_field_and_write_3d_dp(), mo_restart::unpack_field_and_write_3d_dp(), mo_mrm_init::variables_alloc_routing(), mo_mrm_write::write_configfile(), mo_common_restart::write_grid_info(), and mo_restart::write_restart_files().

15.26.2.2 dpc

```
integer, parameter mo_kind::dpc = c_double_complex
```

Double Precision Complex Kind.

Referenced by mo_corr::four1_dp(), mo_corr::four1_sp(), mo_corr::fourrow_dp(), mo_corr::fourrow_sp(), mo_corr::realft_dp(), and mo_corr::zroots_unity_dp().

15.26.2.3 i1

```
integer, parameter mo_kind::i1 = SELECTED_INT_KIND(2)
```

1 Byte Integer Kind

15.26.2.4 i2

```
integer, parameter mo_kind::i2 = c_short
```

2 Byte Integer Kind

15.26.2.5 i4

```
integer, parameter mo_kind::i4 = c_int
```

4 Byte Integer Kind

Referenced by mo_mrm_routing::add_inflow(), mo_julian::caldatjulian(), mo_sce::cce(), mo_sce::chkcst(), mo_meteo_forcings::chunk_config(), mo_meteo_forcings::chunk_size(), mo_sce::comp(), mo_string_utils::compress(),

mo_mrm_init::config_output(), mo_anneal::dchange_dp(), mo_dds::dds(), mo_julian::dec2date360(), mo_julian::dec2date365(), mo_julian::dec2datejulian(), mo_read_forcing_nc::get_time_vector_and_select(), mo_sce::getpnt(), mo_anneal::gettemperature_dp(), mo_grid::init_lowres_level(), mo_meteo_forcings::is_read(), mo_mrm_init::l0_check_input_routing(), mo_mrm_net_startup::l1_l1_mapping(), mo_neutrons::lookupintegral(), mo_dds::mdds(), mhm_driver(), mo_mhm_eval::mhm_eval(), mo_startup::mhm_initialize(), mo_mpr_startup::mpr_initialize(), mrm_driver(), mo_mrm_eval::mrm_eval(), mo_mrm_init::mrm_init(), mo_mrm_init::mrm_init_param(), mo_mrm_restart::mrm_read_restart_config(), mo_mrm_init::mrm_update_param(), mo_mrm_write::mrm_write_output_fluxes(), mo_optimization::optimization(), mo_anneal::pargen_anneal_dp(), mo_sce::parstt(), mo_percentile::percentile_0d_dp(), mo_percentile::percentile_0d_sp(), mo_percentile::percentile_1d_dp(), mo_percentile::percentile_1d_sp(), mo_mpr_pet::pet_correctbyasp(), mo_mrm_init::print_startup_message(), mo_read_forcing_nc::read_forcing_nc(), mo_common_restart::read_grid_info(), mo_read_lut::read_lai_lut(), mo_restart::read_restart_states(), mo_read_spatial_data::read_spatial_data_ascii_i4(), mo_read_forcing_nc::read_weights_nc(), mo_sce::sce(), mo_sce::sort_matrix(), mo_mpr_restart::unpack_field_and_write_1d_i4(), mo_restart::unpack_field_and_write_1d_i4(), mo_mpr_restart::unpack_field_and_write_2d_dp(), mo_restart::unpack_field_and_write_2d_dp(), mo_mpr_restart::unpack_field_and_write_3d_dp(), mo_restart::unpack_field_and_write_3d_dp(), mo_upscaling_operators::upscale_arithmetic_mean(), mo_upscaling_operators::upscale_harmonic_mean(), mo_upscaling_operators::upscale_p_norm(), mo_mrm_init::variables_alloc_routing(), mo_write_ascii::write_configfile(), mo_mrm_write::write_configfile(), mo_mpr_restart::write_eff_params(), mo_common_restart::write_grid_info(), mo_mpr_restart::write_mpr_restart_files(), and mo_restart::write_restart_files().

15.26.2.6 i8

```
integer, parameter mo_kind::i8 = c_long_long
```

8 Byte Integer Kind

Referenced by mo_julian::caldatjulian(), mo_sce::cce(), mo_julian::date2decjulian(), mo_anneal::dchange_dp(), mo_dds::dds(), mo_julian::dec2datejulian(), mo_read_forcing_nc::get_time_vector_and_select(), mo_xor4096::get_timeseed_i8_0d(), mo_xor4096::get_timeseed_i8_1d(), mo_sce::getpnt(), mo_anneal::gettemperature_dp(), mo_julian::juldayjulian(), mo_dds::mdds(), mo_dds::neigh_value(), mo_optimization::optimization(), and mo_sce::sce().

15.26.2.7 lgt

```
integer, parameter mo_kind::lgt = KIND(.true.)
```

Logical Kind.

15.26.2.8 sp

```
integer, parameter mo_kind::sp = c_float
```

Single Precision Real Kind.

Referenced by mo_spatialsimilarity::pd_sp().

15.26.2.9 spc

```
integer, parameter mo_kind::spc = c_float_complex
```

Single Precision Complex Kind.

Referenced by `mo_corr::four1_sp()`, `mo_corr::fourrow_sp()`, `mo_corr::realft_sp()`, and `mo_corr::zroots_unity_sp()`.

15.27 mo_linfit Module Reference

Fitting a straight line.

Data Types

- interface [linfit](#)

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

Functions/Subroutines

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

15.27.1 Detailed Description

Fitting a straight line.

This module provides a routine to fit a straight line with model I or model II regression.

Authors

Matthias Cuntz

Date

Mar 2011

15.27.2 Function/Subroutine Documentation

15.27.2.1 linfit_dp()

```
real(dp) function, dimension(:), allocatable mo_linfit::linfit_dp (
    real(dp), dimension(:), intent(in) x,
    real(dp), dimension(:), intent(in) y,
    real(dp), intent(out), optional a,
    real(dp), intent(out), optional b,
    real(dp), intent(out), optional siga,
    real(dp), intent(out), optional sigb,
    real(dp), intent(out), optional chi2,
    logical, intent(in), optional model2 ) [private]
```

15.27.2.2 linfit_sp()

```
real(sp) function, dimension(:), allocatable mo_linfit::linfit_sp (
    real(sp), dimension(:), intent(in) x,
```

```

real(sp), dimension(:), intent(in) y,
real(sp), intent(out), optional a,
real(sp), intent(out), optional b,
real(sp), intent(out), optional siga,
real(sp), intent(out), optional sigb,
real(sp), intent(out), optional chi2,
logical, intent(in), optional model2 ) [private]

```

15.28 mo_mcmc Module Reference

This module is Monte Carlo Markov Chain sampling of a posterior parameter distribution.

Data Types

- interface [mcmc](#)

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

- interface [mcmc_stddev](#)

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

Functions/Subroutines

- subroutine [mcmc_dp](#) (eval, likelihood, para, rangePar, mcmc_paras, burnin_paras, seed_in, printflag_in, maskpara_in, restart, restart_file, tmp_file, loglike_in, ParaSelectMode_in, iter_burnin_in, iter_mcmc_in, chains_in, stepsize_in)
- subroutine [mcmc_stddev_dp](#) (eval, likelihood, para, rangePar, mcmc_paras, burnin_paras, seed_in, printflag_in, maskpara_in, tmp_file, loglike_in, ParaSelectMode_in, iter_burnin_in, iter_mcmc_in, chains_in, stepsize_in)
- real(dp) function [pargen_dp](#) (old, dMax, oMin, oMax, RN, inbound)
- real(dp) function [pargennorm_dp](#) (old, dMax, oMin, oMax, RN, inbound)
- recursive subroutine [generatenewparameterset_dp](#) (ParaSelectMode, paraold, truepara, rangePar, stepsize, save_state_2, save_state_3, paranew, ChangePara)

15.28.1 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

15.28.2 Function/Subroutine Documentation

15.28.2.1 generateparameterset_dp()

```
recursive subroutine mo_mcmc::generateparameterset_dp (
    integer(i4), intent(in) ParaSelectMode,
    real(dp), dimension(:), intent(in) paraold,
    integer(i4), dimension(:), intent(in) truepara,
    real(dp), dimension(:, :, ), intent(in) rangePar,
    real(dp), dimension(:, ), intent(in) stepsize,
    integer(i8), dimension(n_save_state), intent(inout) save_state_2,
    integer(i8), dimension(n_save_state), intent(inout) save_state_3,
    real(dp), dimension(size(paraold)), intent(out) paranew,
    logical, dimension(size(paraold)), intent(out) ChangePara )
```

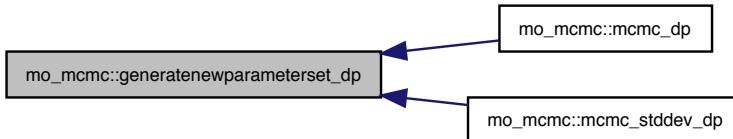
References `pargennorm_dp()`.

Referenced by `mcmc_dp()`, and `mcmc_stddev_dp()`.

Here is the call graph for this function:



Here is the caller graph for this function:



15.28.2.2 mcmc_dp()

```
subroutine mo_mcmc::mcmc_dp (
    procedure(eval_interface), intent(in), pointer eval,
    procedure(objective_interface), intent(in), pointer likelihood,
    real(dp), dimension(:, ), intent(in) para,
    real(dp), dimension(:, :, ), intent(in) rangePar,
    real(dp), dimension(:, :, ), intent(out), allocatable mcmc_paras,
    real(dp), dimension(:, :, ), intent(out), allocatable burnin_paras,
    integer(i8), intent(in), optional seed_in,
    logical, intent(in), optional printflag_in,
    logical, dimension(size(para, 1)), intent(in), optional maskpara_in,
    logical, intent(in), optional restart,
```

```

character(len = *), intent(in), optional restart_file,
character(len = *), intent(in), optional tmp_file,
logical, intent(in), optional loglike_in,
integer(i4), intent(in), optional ParaSelectMode_in,
integer(i4), intent(in), optional iter_burnin_in,
integer(i4), intent(in), optional iter_mcmc_in,
integer(i4), intent(in), optional chains_in,
real(dp), dimension(size(para, 1)), intent(in), optional stepsize_in ) [private]

```

References generateparameterset_dp(), and mo_xor4096::n_save_state.

Here is the call graph for this function:



15.28.2.3 mcmc_stddev_dp()

```

subroutine mo_mcmc::mcmc_stddev_dp (
    procedure(eval_interface), intent(in), pointer eval,
    procedure(objective_interface), intent(in), pointer likelihood,
    real(dp), dimension(:, ), intent(in) para,
    real(dp), dimension(:, :, ), intent(in) rangePar,
    real(dp), dimension(:, :, ), intent(out), allocatable mcmc_paras,
    real(dp), dimension(:, :, ), intent(out), allocatable burnin_paras,
    integer(i8), intent(in), optional seed_in,
    logical, intent(in), optional printflag_in,
    logical, dimension(size(para, 1)), intent(in), optional maskpara_in,
    character(len = *), intent(in), optional tmp_file,
    logical, intent(in), optional loglike_in,
    integer(i4), intent(in), optional ParaSelectMode_in,
    integer(i4), intent(in), optional iter_burnin_in,
    integer(i4), intent(in), optional iter_mcmc_in,
    integer(i4), intent(in), optional chains_in,
    real(dp), dimension(size(para, 1)), intent(in), optional stepsize_in ) [private]

```

References generateparameterset_dp(), and mo_xor4096::n_save_state.

Here is the call graph for this function:



15.28.2.4 pargen_dp()

```
real(dp) function mo_mcmc::pargen_dp (
    real(dp), intent(in) old,
    real(dp), intent(in) dMax,
    real(dp), intent(in) oMin,
    real(dp), intent(in) oMax,
    real(dp), intent(in) RN,
    logical, intent(out), optional inbound ) [private]
```

References mo_kind::dp.

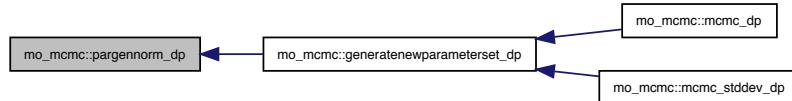
15.28.2.5 pargennorm_dp()

```
real(dp) function mo_mcmc::pargennorm_dp (
    real(dp), intent(in) old,
    real(dp), intent(in) dMax,
    real(dp), intent(in) oMin,
    real(dp), intent(in) oMax,
    real(dp), intent(in) RN,
    logical, intent(out), optional inbound )
```

References mo_kind::dp.

Referenced by generateparameterset_dp().

Here is the caller graph for this function:



15.29 mo_message Module Reference

Write out concatenated strings.

Functions/Subroutines

- subroutine, public **message** (t01, t02, t03, t04, t05, t06, t07, t08, t09, t10, uni, advance)
Write out several string concatenated either on screen or in a file.

Variables

- character(len=1024), public **message_text** = "

15.29.1 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

15.29.2 Function/Subroutine Documentation

15.29.2.1 message()

```
subroutine, public mo_message::message (
    character(len = *), intent(in), optional t01,
    character(len = *), intent(in), optional t02,
    character(len = *), intent(in), optional t03,
    character(len = *), intent(in), optional t04,
    character(len = *), intent(in), optional t05,
    character(len = *), intent(in), optional t06,
    character(len = *), intent(in), optional t07,
    character(len = *), intent(in), optional t08,
    character(len = *), intent(in), optional t09,
    character(len = *), intent(in), optional t10,
    integer, intent(in), optional uni,
    character(len = *), intent(in), optional advance )
```

Write out several string concatenated either on screen or in a file.

Parameters

in	character(len=*) <i>, optional :: t01</i>	1st string
in	character(len=*) <i>, optional :: t02</i>	2nd string
in	character(len=*) <i>, optional :: t03</i>	3rd string
in	character(len=*) <i>, optional :: t04</i>	4th string
in	character(len=*) <i>, optional :: t05</i>	5th string
in	character(len=*) <i>, optional :: t06</i>	6th string
in	character(len=*) <i>, optional :: t07</i>	7th string
in	character(len=*) <i>, optional :: t08</i>	8th string
in	character(len=*) <i>, optional :: t09</i>	9th string
in	character(len=*) <i>, optional :: t10</i>	10th string
in	integer <i>, optional :: unit</i>	Unit to write to (default: nout)
in	character(len=*) <i>, optional :: advance</i>	WRITE advance keyword (default: 'yes') yes: newline will be written after message no: no newline at end of message

Author

Matthias Cuntz - modified from Echam5, (C) MPI-MET, Hamburg, Germany

Date

Dec 2011

References `mo_constants::nout`.

Referenced by `mo_grid::calculate_grid_properties()`, `mo_multi_param_reg::canopy_intercept_param()`, `mo_read_wrapper::check_consistency_lut_map()`, `mo_common_mhm_mrm_read_config::check_optimization_settings()`, `mo_meteo_forcings::chunk_size()`, `mo_mrm_write_fluxes_states::close()`, `mo_write_fluxes_states::close()`, `mo_common_mhm_mrm_read_config::common_check_resolution()`, `mo_common_mhm_mrm_read_config::common_mhm_mrm_read_config()`, `mo_common_read_config::common_read_config()`, `mo_mrm_init::config_output()`, `mo_startup::constants_init()`, `mo_objective_function::extract_basin_avg_tws()`, `mo_mrm_objective_function_runoff::extract_runoff()`, `mo_soil_database::generate_soil_database()`, `mo_read_forcing_nc::get_time_vector_and_select()`, `mo_meteo_forcings::is_read()`, `mo_mpr_startup::l0_check_input()`, `mo_mrm_init::l0_check_input_routing()`, `mo_mrm_net_startup::l11_flow_direction()`, `mo_mrm_net_startup::l11_link_location()`, `mo_startup::l2_variable_init()`, `mo_mrm_signatures::limb_densities()`, `mo_mrm_signatures::maximummonthlyflow()`, `mo_mhm::mhm()`, `mhm_driver()`, `mo_mhm_eval::mhm_eval()`, `mo_mhm_read_config::mhm_read_config()`, `mo_mrm_signatures::moments()`, `mo_multi_param_reg::mpr()`, `mo_mpr_eval::mpr_eval()`, `mo_mpr_read_config::mpr_read_config()`, `mo_mpr_soilmoist::mpr_sm()`, `mo_mpr_smhorizons::mpr_smhorizons()`, `mrm_driver()`, `mo_mrm_eval::mrm_eval()`, `mo_mrm_init::mrm_init()`, `mo_mrm_init::mrm_init_param()`, `mo_mrm_read_config::mrm_read_config()`, `mo_mrm_read_data::mrm_read_discharge()`, `mo_mrm_read_data::mrm_read_l0_data()`, `mo_mrm_restart::mrm_read_restart_config()`, `mo_mrm_write::mrm_write_optifile()`, `mo_mrm_write::mrm_write_optinamelist()`, `mo_mrm_restart::mrm_write_restart()`, `mo_objective_function::objective()`, `mo_objective_function::objective_et_kge_catchment_avg()`, `mo_objective_function::objective_kge_q_et()`, `mo_objective_function::objective_kge_q_rmse_et()`, `mo_objective_function::objective_kge_q_rmse_tws()`, `mo_objective_function::objective_kge_q_sm_corr()`, `mo_objective_function::objective_neutrons_kge_catchment_avg()`, `mo_objective_function::objective_sm_corr()`, `mo_objective_function::objective_sm_kge_catchment_avg()`, `mo_objective_function::objective_sm_pd()`, `mo_objective_function::objective_sm_sse_standard_score()`, `mo_nml::open_nml()`, `mo_optimization::optimization()`, `mo_prepare_gridded_lai::prepare_gridded_daily_lai_data()`, `mo_prepare_gridded_lai::prepare_gridded_mean_monthly_lai_data()`, `mo_meteo_forcings::prepare_meteo_forcings_data()`, `mo_mrm_init::print_startup_message()`, `mo_read_optional_data::read_basin_avg_tws()`, `mo_read_wrapper::read_data()`, `mo_common_read_data::read_dem()`, `mo_read_optional_data::read_evapotranspiration()`, `mo_read_forcing_nc::read_forcing_nc()`, `mo_common_restart::read_grid_info()`, `mo_read_latlon::read_latlon()`, `mo_common_read_data::read_lcover()`, `mo_mrm_read_config::read_mrm_routing_params()`, `mo_read_optional_data::read_neutrons()`, `mo_soil_database::read_soil_lut()`, `mo_read_optional_data::read_soil_moisture()`, `mo_read_timeseries::read_timeseries()`, `mo_read_forcing_nc::read_weights_nc()`, `mo_mrm_signatures::runoffratio()`, `mo_common_read_config::set_land_cover_scenes_id()`, `mo_mrm_objective_function_runoff::single_objective_runoff()`, `mo_write_ascii::write_configfile()`, `mo_mrm_write::write_configfile()`, `mo_mrm_write::write_daily_obs_sim_discharge()`, `mo_mpr_restart::write_mpr_restart_files()`, `mo_write_ascii::write_optifile()`, `mo_write_ascii::write_optinamelist()`, `mo_restart::write_restart_files()`, and `mo_mrm_signatures::zeroflowratio()`.

15.29.3 Variable Documentation

15.29.3.1 `message_text`

```
character(len = 1024), public mo_message::message_text = ''
```

Referenced by `mo_mpr_startup::l0_check_input()`, `mo_mrm_init::l0_check_input_routing()`, `mhm_driver()`, `mo_nml::open_nml()`, `mo_nml::position_nml()`, and `mo_mrm_init::print_startup_message()`.

15.30 `mo_meteo_forcings` Module Reference

Prepare meteorological forcings data for mHM.

Functions/Subroutines

- subroutine, public `prepare_meteo_forcings_data` (iBasin, tt)
Prepare meteorological forcings data for a given variable.
- subroutine `meteo_forcings_wrapper` (iBasin, dataPath, inputFormat, dataOut1, lower, upper, ncvarName)
Prepare meteorological forcings data for mHM at Level-1.
- subroutine `meteo_weights_wrapper` (iBasin, read_meteo_weights, dataPath, dataOut1, lower, upper, ncvarName)
Prepare weights for meteorological forcings data for mHM at Level-1.
- subroutine `chunk_config` (iBasin, tt, read_flag, readPer)
determines the start date, end date, and read_flag given basin id and current timestep
- logical function `is_read` (iBasin, tt)
evaluate whether new chunk should be read at this timestep
- subroutine `chunk_size` (iBasin, tt, readPer)
calculate beginning and end of read Period, i.e. that is length of current chunk to read

15.30.1 Detailed Description

Prepare meteorological forcings data for mHM.

Prepare meteorological forcings data for mHM.

Authors

Rohini Kumar

Date

Jan 2012

15.30.2 Function/Subroutine Documentation

15.30.2.1 `chunk_config()`

```
subroutine mo_meteo_forcings::chunk_config (
    integer(i4), intent(in) iBasin,
    integer(i4), intent(in) tt,
    logical, intent(out) read_flag,
    type(period), intent(out) readPer )
```

determines the start date, end date, and read_flag given basin id and current timestep

TODO: add description

Parameters

in	<code>integer(i4) :: iBasin</code>	current Basin
in	<code>integer(i4) :: tt</code>	current timestep
out	<code>logical :: read_flag</code>	indicate whether reading data should be read
out	<code>type(period) :: readPer</code>	start and end dates of reading Period

Authors

Stephan Thober

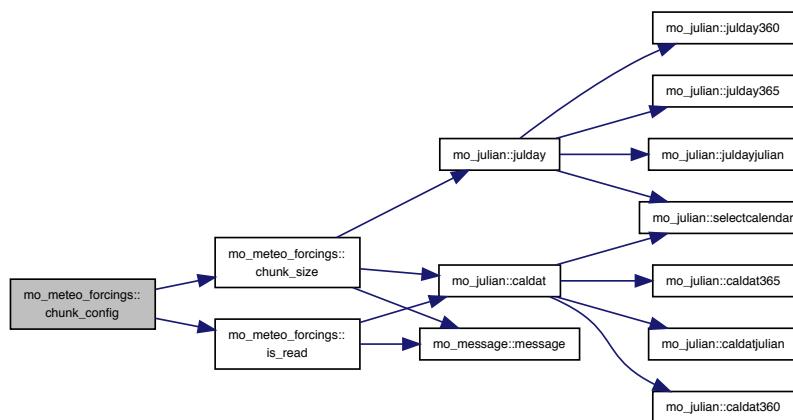
Date

Jun 2014

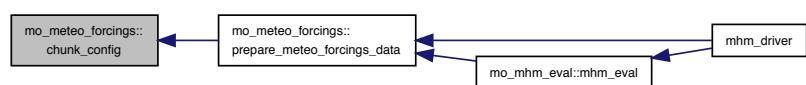
References `chunk_size()`, `mo_kind::i4`, `is_read()`, and `mo_common_constants::nodata_dp`.

Referenced by `prepare_meteo_forcings_data()`.

Here is the call graph for this function:



Here is the caller graph for this function:



15.30.2.2 `chunk_size()`

```
subroutine mo_meteo_forcings::chunk_size (
    integer(i4), intent(in) iBasin,
    integer(i4), intent(in) tt,
    type(period), intent(out) readPer )
```

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

TODO: add description

Parameters

in	<code>integer(i4) :: iBasin</code>	current Basin to process
----	------------------------------------	--------------------------

Parameters

in	<i>integer(i4) :: tt</i>	current time step
out	<i>type(period) :: readPer</i>	start and end dates of read Period

Authors

Stephan Thober

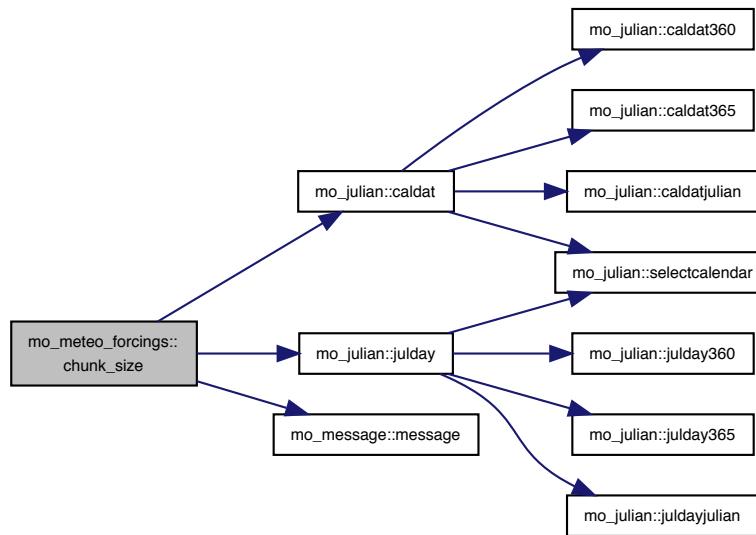
Date

Jun 2014

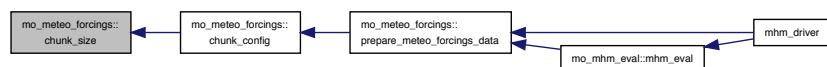
References `mo_julian::caldat()`, `mo_kind::i4`, `mo_julian::julday()`, `mo_message::message()`, `mo_common_mhm_mrm_variables::ntstepday`, `mo_common_mhm_mrm_variables::simper`, and `mo_global_variables::timestep_model_inputs`.

Referenced by `chunk_config()`.

Here is the call graph for this function:



Here is the caller graph for this function:



15.30.2.3 is_read()

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

evaluate whether new chunk should be read at this timestep

TODO: add description

Parameters

in	<i>integer(i4) :: iBasin</i>	current Basin
in	<i>integer(i4) :: tt</i>	current time step

Authors

Stephan Thober

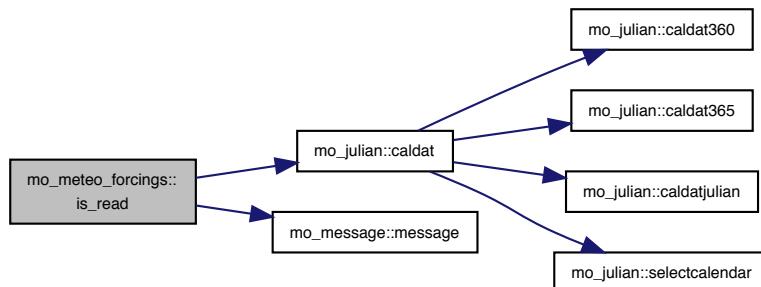
Date

Jun 2014

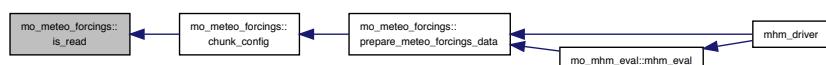
References `mo_julian::caldat()`, `mo_kind::i4`, `mo_message::message()`, `mo_common_mhm_mrm_variables::timestepday`, `mo_common_mhm_mrm_variables::simper`, `mo_common_mhm_mrm_variables::timestep`, and `mo_global_variables::timestep_model_inputs`.

Referenced by `chunk_config()`.

Here is the call graph for this function:



Here is the caller graph for this function:



15.30.2.4 meteo_forcings_wrapper()

```
subroutine mo_meteo_forcings::meteo_forcings_wrapper (
    integer(i4), intent(in) iBasin,
    character(len = *), intent(in) dataPath,
    character(len = *), intent(in) inputFormat,
    real(dp), dimension(:, :, ), intent(inout), allocatable dataOut1,
    real(dp), intent(in), optional lower,
    real(dp), intent(in), optional upper,
    character(len = *), intent(in), optional ncvarName )
```

Prepare meteorological forcings data for mHM at Level-1.

Prepare meteorological forcings data for mHM, which include 1) Reading meteo. datasets at their native resolution for every basin 2) Perform aggregation or disaggregation of meteo. datasets from their native resolution (level-2) to the required hydrologic resolution (level-1) 3) Pad the above datasets of every basin to their respective global ones

Parameters

in	integer(i4) :: iBasin	Basin Id
in	character(len = *) :: dataPath	Data path where a given meteo. variable is stored
in	character(len = *), intent(in) :: inputFormat	only 'nc' possible at the moment
in, out	real(dp), dimension(:, :,) :: dataOut1	Packed meterological variable for the whole simulation period
in	real(dp), optional :: lower	Lower bound for check of validity of data values
in	real(dp), optional :: upper	Upper bound for check of validity of data values
in	character(len = *), optional :: ncvarName	name of the variable (for .nc files)

Authors

Rohini Kumar

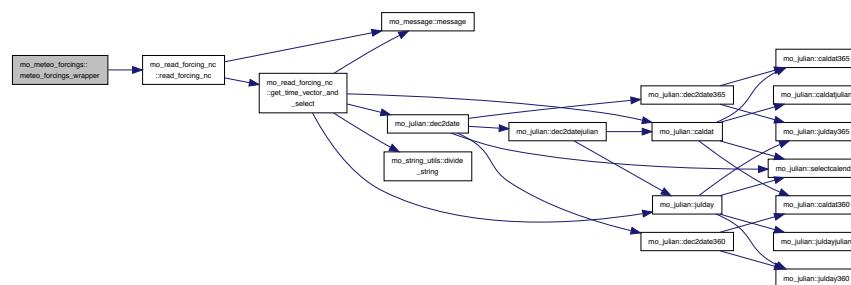
Date

Jan 2013

References mo_common_variables::level1, mo_global_variables::level2, mo_common_constants::nodata_dp, mo_read_forcing_nc::read_forcing_nc(), and mo_common_mhm_mrm_variables::readper.

Referenced by prepare_meteo_forcings_data().

Here is the call graph for this function:



Here is the caller graph for this function:



15.30.2.5 meteo_weights_wrapper()

```

subroutine mo_meteo_forcings::meteo_weights_wrapper (
    integer(i4), intent(in) iBasin,
    logical, intent(in) read_meteo_weights,
    character(len = *), intent(in) dataPath,
    real(dp), dimension(:, :, :), intent(inout), allocatable dataOut1,
    real(dp), intent(in), optional lower,
    real(dp), intent(in), optional upper,
    character(len = *), intent(in), optional ncvarName )
  
```

Prepare weights for meteorological forcings data for mHM at Level-1.

Prepare meteorological weights data for mHM, which include 1) Reading meteo. weights datasets at their native resolution for every basin 2) Perform aggregation or disaggregation of meteo. weights datasets from their native resolution (level-2) to the required hydrologic resolution (level-1) 3) Pad the above datasets of every basin to their respective global ones

Parameters

in	integer(i4) :: iBasin	Basin Id
in	logical :: read_meteo_weights	Flag for reading meteo weights
in	character(len = *) :: dataPath	Data path where a given meteo. variable is stored
in, out	real(dp), dimension(:, :, :) :: dataOut1	Packed meterological variable for the whole simulation period
in	real(dp), optional :: lower	Lower bound for check of validity of data values
in	real(dp), optional :: upper	Upper bound for check of validity of data values
in	character(len = *), optional :: ncvarName	name of the variable (for .nc files)

Authors

Stephan Thober & Rohini Kumar

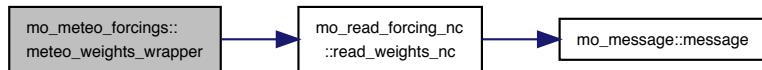
Date

Jan 2017

References mo_common_variables::level1, mo_global_variables::level2, mo_common_constants::nodata_dp, and mo_read_forcing_nc::read_weights_nc().

Referenced by prepare_meteo_forcings_data().

Here is the call graph for this function:



Here is the caller graph for this function:



15.30.2.6 prepare_meteo_forcings_data()

```
subroutine, public mo_meteo_forcings::prepare_meteo_forcings_data (
    integer(i4), intent(in) iBasin,
    integer(i4), intent(in) tt )
```

Prepare meteorological forcings data for a given variable.

Prepare meteorological forcings data for a given variable. Internally this subroutine calls another routine meteo_wrapper for different meterological variables

Parameters

in	integer(i4) :: iBasin	Basin Id
in	integer(i4) :: tt	current timestep

Authors

Rohini Kumar

Date

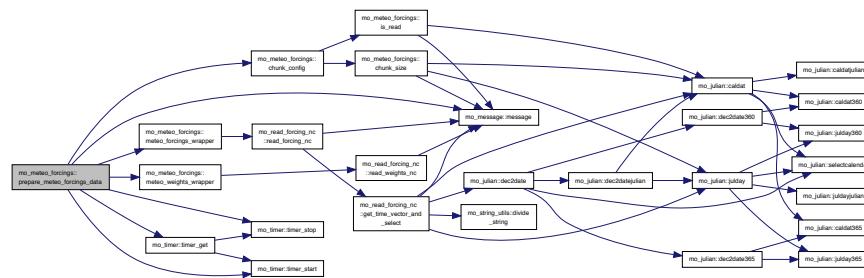
Jan 2013

References chunk_config(), mo_global_variables::dirabsvappressure, mo_global_variables::dirmaxtemperature, mo_global_variables::dirmintemperature, mo_global_variables::dirnetradiation, mo_global_variables::dirprecipitation, mo_global_variables::dirreferenceet, mo_global_variables::dirtemperature, mo_global_variables::dirwindspeed, mo_global_variables::inputformat_meteo_forcings, mo_global_variables::l1_absvappress, mo_global_variables::l1_netrad, mo_global_variables::l1_pet, mo_global_variables::l1_pet_weights, mo_global_variables::l1_pre, mo_global_variables::l1_pre_weights, mo_global_variables::l1_temp, mo_global_variables::l1_temp_weights, mo_global_variables::l1_tmax, mo_global_variables::l1_tmin, mo_global_variables::l1_windspeed, mo_message::message(), meteo_forcings_wrapper(), meteo_weights_wrapper(), mo_common_variables::nbasins, mo

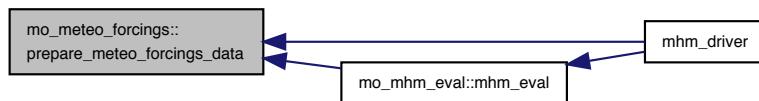
`_common_variables::processmatrix`, `mo_global_variables::read_meteo_weights`, `mo_common_mhm_mrm::variables::readper`, `mo_timer::timer_get()`, `mo_timer::timer_start()`, `mo_timer::timer_stop()`, and `mo_global_variables::timestep_model_inputs`.

Referenced by `mhm_driver()`, and `mo_mhm_eval::mhm_eval()`.

Here is the call graph for this function:



Here is the caller graph for this function:



15.31 mo_mhm Module Reference

Call all main processes of mHM.

Functions/Subroutines

- subroutine, public `mhm` (`read_states`, `tt`, `time`, `processMatrix`, `horizon_depth`, `nCells1`, `nHorizons_mHM`, `ntimesteps_day`, `neutron_integral_AFast`, `global_parameters`, `latitude`, `evap_coeff`, `fdays_prec`, `fnights_prec`, `fdays_pet`, `fnights_pet`, `fdays_temp`, `fnights_temp`, `temp_weights`, `pet_weights`, `pre_weights`, `read_meteo_weights`, `pet_in`, `tmin_in`, `tmax_in`, `netrad_in`, `absvappres_in`, `windspeed_in`, `prec_in`, `temp_in`, `fSealed1`, `interc`, `snowpack`, `sealedStorage`, `soilMoisture`, `unsatStorage`, `satStorage`, `neutrons`, `pet_calc`, `aet_soil`, `aet_canopy`, `aet_sealed`, `baseflow`, `infiltration`, `fast_interflow`, `melt`, `perc`, `prec_effect`, `rain`, `runoff_sealed`, `slow_interflow`, `snow`, `throughfall`, `total_runoff`, `alpha`, `deg_day_incr`, `deg_day_max`, `deg_day_noprec`, `deg_day_fAsp`, `petLAlcorFactorL1`, `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.

15.31.1 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

15.31.2 Function/Subroutine Documentation

15.31.2.1 mhm()

```
subroutine, public mo_mhm::mhm (
    logical, intent(in) read_states,
    integer(i4), intent(in) tt,
    real(dp), intent(in) time,
    integer(i4), dimension(:, :), intent(in) processMatrix,
    real(dp), dimension(:, :), intent(in) horizon_depth,
    integer(i4), intent(in) nCells1,
    integer(i4), intent(in) nHorizons_mHM,
```

```
real(dp), intent(in) ntimesteps_day,
real(dp), 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	<i>logical :: read_states</i>	indicated whether states have been read from file
in	<i>integer(i4) :: tt</i>	simulation time step
in	<i>real(dp) :: time</i>	current decimal Julian day
in	<i>integer(i4), dimension(:, :) :: processMatrix</i>	mHM process configuration matrix
in	<i>real(dp), dimension(:,) :: horizon_depth</i>	Depth of each horizon in mHM
in	<i>integer(i4) :: nCells1</i>	number of cells in a given basin at level L1
in	<i>integer(i4) :: nHorizons_mHM</i>	Number of Horizons in mHM
in	<i>real(dp) :: ntimesteps_day</i>	number of time intervals per day, transformed in dp
in	<i>real(dp), dimension(:,) :: neutron_integral_AFast</i>	tabular for neutron flux approximation
in	<i>real(dp), dimension(:,) :: global_parameters</i>	global mHM parameters
in	<i>real(dp), dimension(:,) :: latitude</i>	latitude on level 1
in	<i>real(dp), dimension(:,) :: evap_coeff</i>	Evaporation coefficient for free-water surface of that current month
in	<i>real(dp), dimension(:,) :: fday_prec</i>	[-] day ratio precipitation < 1
in	<i>real(dp), dimension(:,) :: fnight_prec</i>	[-] night ratio precipitation < 1
in	<i>real(dp), dimension(:,) :: fday_pet</i>	[-] day ratio PET < 1
in	<i>real(dp), dimension(:,) :: fnight_pet</i>	[-] night ratio PET < 1
in	<i>real(dp), dimension(:,) :: fday_temp</i>	[-] day factor mean temp
in	<i>real(dp), dimension(:,) :: fnight_temp</i>	[-] night factor mean temp
in	<i>real(dp), dimension(:, :, :) :: temp_weights</i>	multiplicative weights for temperature (deg K)
in	<i>real(dp), dimension(:, :, :) :: pet_weights</i>	multiplicative weights for potential evapotranspiration
in	<i>real(dp), dimension(:, :, :) :: pre_weights</i>	multiplicative weights for precipitation
in	<i>logical :: read_meteo_weights</i>	flag whether weights for tavg and pet have read and should be used
in	<i>real(dp), dimension(:,) :: pet_in</i>	[mm d-1] Daily potential evapotranspiration (input)
in	<i>real(dp), dimension(:,) :: tmin_in</i>	[degc] Daily minimum temperature
in	<i>real(dp), dimension(:,) :: tmax_in</i>	[degc] Daily maximum temperature
in	<i>real(dp), dimension(:,) :: netrad_in</i>	[w m ²] Daily average net radiation

Parameters

in	<i>real(dp)</i> , dimension(:) :: <i>absvappres_in</i>	[Pa] Daily average absolute vapour pressure
in	<i>real(dp)</i> , dimension(:) :: <i>windspeed_in</i>	[m s-1] Daily average wind speed
in	<i>real(dp)</i> , dimension(:) :: <i>prec_in</i>	[mm d-1] Daily mean precipitation
in	<i>real(dp)</i> , dimension(:) :: <i>temp_in</i>	[degc] Daily average temperature
in, out	<i>real(dp)</i> , dimension(:) :: <i>fSealed1</i>	fraction of sealed area at scale L1
in, out	<i>real(dp)</i> , dimension(:) :: <i>interc</i>	Interception
in, out	<i>real(dp)</i> , dimension(:) :: <i>snowpack</i>	Snowpack
in, out	<i>real(dp)</i> , dimension(:) :: <i>sealedStorage</i>	Retention storage of impervious areas
in, out	<i>real(dp)</i> , dimension(:, :) :: <i>soilMoisture</i>	Soil moisture of each horizon
in, out	<i>real(dp)</i> , dimension(:) :: <i>unsatStorage</i>	Upper soil storage
in, out	<i>real(dp)</i> , dimension(:) :: <i>satStorage</i>	Groundwater storage
in, out	<i>real(dp)</i> , dimension(:) :: <i>neutrons</i>	Ground albedo neutrons
in, out	<i>real(dp)</i> , dimension(:) :: <i>pet_calc</i>	[mm TST-1] estimated PET (if PET is input = corrected values (fAsp*PET))
in, out	<i>real(dp)</i> , dimension(:, :) :: <i>aet_soil</i>	actual ET
in, out	<i>real(dp)</i> , dimension(:) :: <i>aet_canopy</i>	Real evaporation intensity from canopy
in, out	<i>real(dp)</i> , dimension(:) :: <i>aet_sealed</i>	Actual ET from free-water surfaces
in, out	<i>real(dp)</i> , dimension(:) :: <i>baseflow</i>	Baseflow
in, out	<i>real(dp)</i> , dimension(:, :) :: <i>infiltration</i>	Recharge, infiltration intensity or effective precipitation of each horizon
in, out	<i>real(dp)</i> , dimension(:) :: <i>fast_interflow</i>	Fast runoff component
in, out	<i>real(dp)</i> , dimension(:) :: <i>melt</i>	Melting snow depth
in, out	<i>real(dp)</i> , dimension(:) :: <i>perc</i>	Percolation
in, out	<i>real(dp)</i> , dimension(:) :: <i>prec_effect</i>	Effective precipitation depth (snow melt + rain)
in, out	<i>real(dp)</i> , dimension(:) :: <i>rain</i>	Rain precipitation depth
in, out	<i>real(dp)</i> , dimension(:) :: <i>runoff_sealed</i>	Direct runoff from impervious areas
in, out	<i>real(dp)</i> , dimension(:) :: <i>slow_interflow</i>	Slow runoff component
in, out	<i>real(dp)</i> , dimension(:) :: <i>snow</i>	Snow precipitation depth
in, out	<i>real(dp)</i> , dimension(:) :: <i>throughfall</i>	Throughfall
in, out	<i>real(dp)</i> , dimension(:) :: <i>total_runoff</i>	Generated runoff
in, out	<i>real(dp)</i> , dimension(:) :: <i>alpha</i>	Exponent for the upper reservoir
in, out	<i>real(dp)</i> , dimension(:) :: <i>deg_day_incr</i>	Increase of the Degree-day factor per mm of increase in precipitation
in, out	<i>real(dp)</i> , dimension(:) :: <i>deg_day_max</i>	Maximum Degree-day factor
in, out	<i>real(dp)</i> , dimension(:) :: <i>deg_day_noprec</i>	Degree-day factor with no precipitation
in, out	<i>real(dp)</i> , dimension(:) :: <i>deg_day</i>	Degree-day factor
in, out	<i>real(dp)</i> , dimension(:) :: <i>fAsp</i>	[1] PET correction for Aspect at level 1
in, out	<i>real(dp)</i> , dimension(:) :: <i>petLAIcorFactorL1</i>	PET correction factor based on LAI at level 1
in, out	<i>real(dp)</i> , dimension(:) :: <i>HarSamCoeff</i>	[1] PET Hargreaves Samani coefficient at level 1
in, out	<i>real(dp)</i> , dimension(:) :: <i>PrieTayAlpha</i>	[1] PET Priestley Taylor coefficient at level 1
in, out	<i>real(dp)</i> , dimension(:) :: <i>aeroResist</i>	[s m-1] PET aerodynamical resistance at level 1
in, out	<i>real(dp)</i> , dimension(:) :: <i>surfResist</i>	[s m-1] PET bulk surface resistance at level 1
in, out	<i>real(dp)</i> , dimension(:, :) :: <i>frac_roots</i>	Fraction of Roots in soil horizon
in, out	<i>real(dp)</i> , dimension(:) :: <i>interc_max</i>	Maximum interception
in, out	<i>real(dp)</i> , dimension(:) :: <i>karst_loss</i>	Karstic percolation loss

Parameters

in, out	<i>real(dp), dimension(:) :: k0</i>	Recession coefficient of the upper reservoir, upper outlet
in, out	<i>real(dp), dimension(:) :: k1</i>	Recession coefficient of the upper reservoir, lower outlet
in, out	<i>real(dp), dimension(:) :: k2</i>	Baseflow recession coefficient
in, out	<i>real(dp), dimension(:) :: kp</i>	Percolation coefficient
in, out	<i>real(dp), dimension(:, :) :: soil_moist_FC</i>	Soil moisture below which actual ET is reduced
in, out	<i>real(dp), dimension(:, :) :: soil_moist_sat</i>	Saturation soil moisture for each horizon [mm]
in, out	<i>real(dp), dimension(:, :) :: soil_moist_exponen</i>	Exponential parameter to how non-linear is the soil water retention
in, out	<i>real(dp), dimension(:) :: jarvis_thresh_c1</i>	jarvis critical value for normalized soil water content
in, out	<i>real(dp), dimension(:) :: temp_thresh</i>	Threshold temperature for snow/rain
in, out	<i>real(dp), dimension(:) :: unsat_thresh</i>	Threshold water depth in upper reservoir
in, out	<i>real(dp), dimension(:) :: water_thresh_sealed</i>	Threshold water depth in impervious areas
in, out	<i>real(dp), dimension(:, :) :: wilting_point</i>	Permanent wilting point for each horizon

Authors

Luis Samaniego & Rohini Kumar

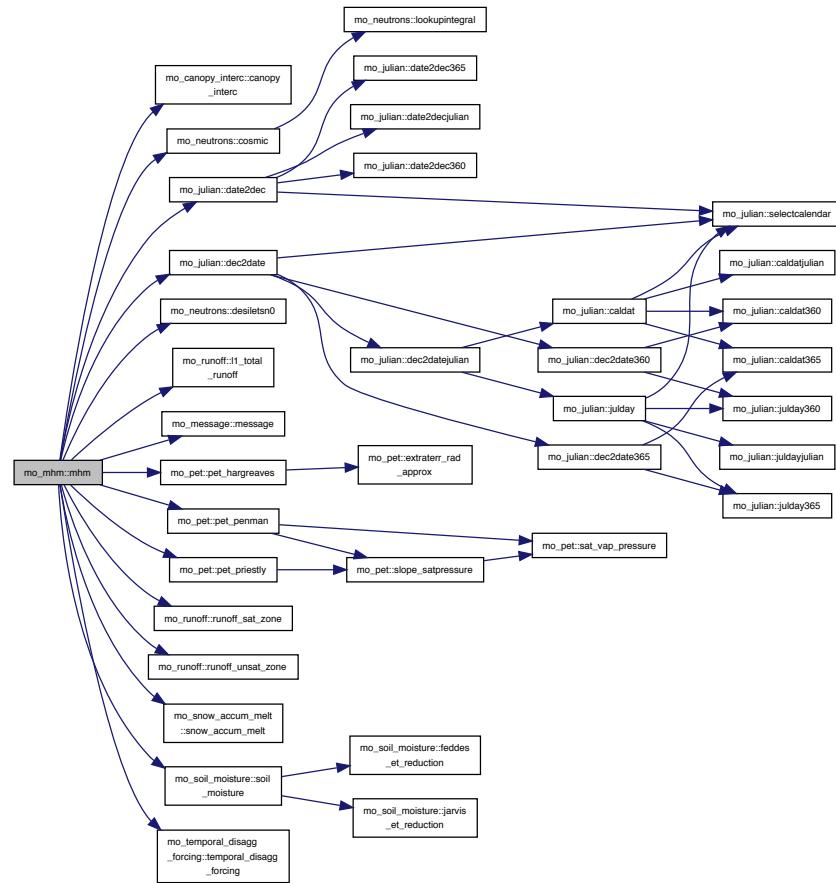
Date

Dec 2012

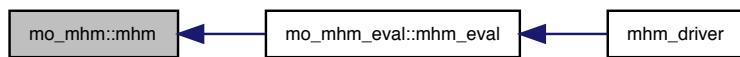
References mo_canopy_interc::canopy_interc(), mo_neutrons::cosmic(), mo_julian::date2dec(), mo_julian::dec2date(), mo_neutrons::desiletsn0(), mo_mhm_constants::harsamconst, mo_runoff::l1_total_runoff(), mo_message::message(), mo_pet::pet_hargreaves(), mo_pet::pet_penman(), mo_pet::pet_priestly(), mo_runoff::runoff_sat_zone(), mo_runoff::runoff_unsat_zone(), mo_snow_accum_melt::snow_accum_melt(), mo_soil::moisture::soil_moisture(), and mo_temporal_disagg_forcing::temporal_disagg_forcing().

Referenced by mo_mhm_eval::mhm_eval().

Here is the call graph for this function:



Here is the caller graph for this function:



15.32 `mo_mhm_constants` Module Reference

Provides mHM specific constants.

Variables

- real(dp), parameter, public `h2odens` = 1000.0_dp
- real(dp), parameter, public `p2_initstatefluxes` = 15.00_dp
- real(dp), parameter, public `p3_initstatefluxes` = 10.00_dp

- real(dp), parameter, public `p4_initstatefluxes` = 75.00_dp
- real(dp), parameter, public `p5_initstatefluxes` = 1500.00_dp
- real(dp), parameter, public `c1_initstatesm` = 0.25_dp
- integer(i4), parameter, public `noutflxstate` = 20_i4
- real(dp), parameter, public `stbolzmann` = 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

15.32.1 Detailed Description

Provides mHM specific constants.

Provides mHM specific constants such as flood plain elevation.

Authors

Matthias Cuntz

Date

Nov 2011

15.32.2 Variable Documentation

15.32.2.1 c1_initstatesm

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

15.32.2.2 cosmic_alpha

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

Referenced by mo_neutrons::cosmic().

15.32.2.3 cosmic_bd

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

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

Referenced by mo_neutrons::cosmic().

15.32.2.4 cosmic_l1

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

Referenced by mo_neutrons::cosmic().

15.32.2.5 cosmic_l2

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

Referenced by mo_neutrons::cosmic().

15.32.2.6 cosmic_l3

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

Referenced by mo_neutrons::cosmic().

15.32.2.7 cosmic_l4

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

Referenced by mo_neutrons::cosmic().

15.32.2.8 cosmic_n

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

Referenced by mo_neutrons::cosmic().

15.32.2.9 cosmic_vwclat

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

Referenced by mo_neutrons::cosmic().

15.32.2.10 desilets_a0

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

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

Referenced by mo_neutrons::desiletsn0().

15.32.2.11 desilets_a1

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

Referenced by mo_neutrons::desiletsn0().

15.32.2.12 desilets_a2

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

Referenced by mo_neutrons::desiletsn0().

15.32.2.13 duffedelta1

```
real(dp), parameter, public mo_mhm_constants::duffedelta1 = 0.4090_dp
```

Referenced by mo_pet::extraterr_rad_approx().

15.32.2.14 duffedelta2

```
real(dp), parameter, public mo_mhm_constants::duffedelta2 = 1.3900_dp
```

Referenced by mo_pet::extraterr_rad_approx().

15.32.2.15 duffedr

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

Referenced by mo_pet::extraterr_rad_approx().

15.32.2.16 h2odens

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

Referenced by mo_neutrons::cosmic().

15.32.2.17 harsamconst

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

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

Referenced by mo_mhm::mhm().

15.32.2.18 noutfluxstate

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

15.32.2.19 p2_initstatefluxes

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

15.32.2.20 p3_initstatefluxes

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

15.32.2.21 p4_initstatefluxes

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

15.32.2.22 p5_initstatefluxes

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

15.32.2.23 satpressureslope1

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

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

Referenced by mo_pet::slope_satpressure().

15.32.2.24 stboltzmann

```
real(dp), parameter, public mo_mhm_constants::stboltzmann = 5.67e-08_dp
```

Stefan-Boltzmann constant [W m^-2 K^-4].

15.32.2.25 tetens_c1

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

Tetens's formula to calculate saturated vapour pressure.

Referenced by mo_pet::sat_vap_pressure().

15.32.2.26 tetens_c2

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

Referenced by mo_pet::sat_vap_pressure().

15.32.2.27 tetens_c3

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

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

15.33 mo_mhm_eval Module Reference

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

Functions/Subroutines

- subroutine, public mhm_eval (parameterset, runoff, sm_opti, basin_avg_tws, neutrons_opti, et_opti)
Runs mhm with a specific parameter set and returns required variables, e.g. runoff.

15.33.1 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

15.33.2 Function/Subroutine Documentation

15.33.2.1 mhm_eval()

```
subroutine, public mo_mhm_eval::mhm_eval (
    real(dp), dimension(:), intent(in) parameterset,
```

```

real(dp), dimension(:, :, ), intent(out), optional, allocatable runoff,
real(dp), dimension(:, :, ), intent(out), optional, allocatable sm_opti,
real(dp), dimension(:, :, ), intent(out), optional, allocatable basin_avg_tws,
real(dp), dimension(:, :, ), intent(out), optional, allocatable neutrons_opti,
real(dp), dimension(:, :, ), intent(out), optional, allocatable et_opti )

```

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

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

Parameters

in	<i>real(dp), dimension(:) :: parameterset</i>	a set of global parameter (gamma) to run mHM, DIMENSION [no. of global_Parameters]
out	<i>real(dp), dimension(:, :,), optional :: runoff</i>	returns runoff time series, DIMENSION [nTimeSteps, nGaugesTotal]
out	<i>real(dp), dimension(:, :,), optional :: sm_opti</i>	returns soil moisture time series for all grid cells (of multiple basins concatenated),DIMENSION [nCells, nTimeSteps]
out	<i>real(dp), dimension(:, :,), optional :: basin_avg_tws</i>	returns basin averaged total water storage time series, DIMENSION [nTimeSteps, nBasins]
out	<i>real(dp), dimension(:, :,), optional :: neutrons_opti</i>	dim1=ncells, dim2=time
out	<i>real(dp), dimension(:, :,), optional :: et_opti</i>	returns evapotranspiration time series for all grid cells (of multiple basins concatenated),DIMENSION [nCells, nTimeSteps]

Authors

Juliane Mai, Rohini Kumar

Date

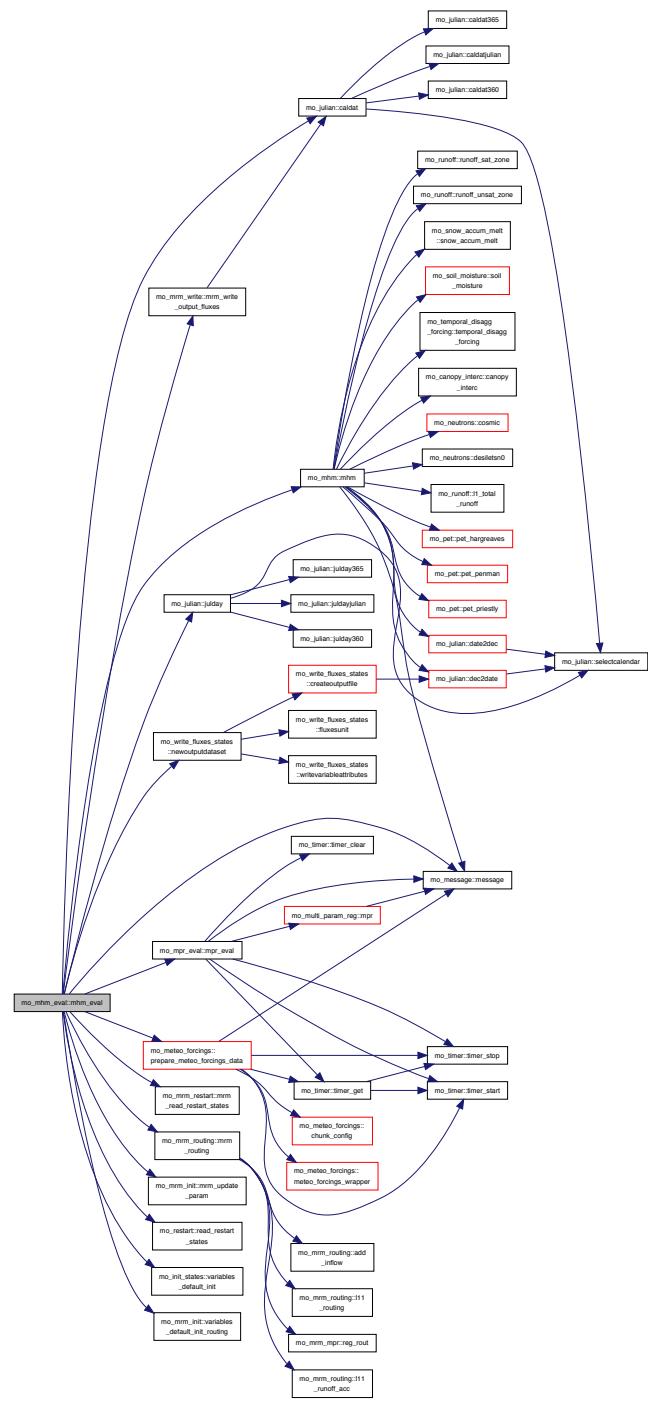
Feb 2013

References mo_global_variables::basin_avg_tws_sim, mo_mrm_global_variables::basin_mrm, mo_julian::caldat(), mo_common_mhm_mrm_variables::dirrestartin, mo_global_variables::evap_coeff, mo_global_variables::fday←_pet, mo_global_variables::fday_prec, mo_global_variables::fday_temp, mo_global_variables::fnight_pet, mo←_global_variables::fnight_prec, mo_global_variables::fnight_temp, mo_mpr_global_variables::horizondepth←mhm, mo_common_constants::hoursecs, mo_kind::i4, mo_mrm_global_variables::inflowgauge, mo_julian←::julday(), mo_mrm_global_variables::i11_c1, mo_mrm_global_variables::i11_c2, mo_mrm_global_variables←::i11_fromn, mo_mrm_global_variables::i11_l1_id, mo_mrm_global_variables::i11_length, mo_mrm_global←variables::i11_netperm, mo_mrm_global_variables::i11_nlinkfracfpimp, mo_mrm_global_variables::i11_noutlets, mo_mrm_global_variables::i11_qmod, mo_mrm_global_variables::i11_qout, mo_mrm_global_variables::i11←_qtin, mo_mrm_global_variables::i11_qtr, mo_mrm_global_variables::i11_slope, mo_mrm_global_variables←::i11_ton, mo_mrm_global_variables::i11_tsrou, mo_global_variables::i1_absvappress, mo_mpr_global←variables::i1_aeroresist, mo_global_variables::i1_aetcany, mo_global_variables::i1_aetsealed, mo_global←variables::i1_aetsoil, mo_mpr_global_variables::i1_alpha, mo_global_variables::i1_baseflow, mo_mpr_global←variables::i1_degday, mo_mpr_global_variables::i1_degdayinc, mo_mpr_global_variables::i1_degdaymax, mo←_mpr_global_variables::i1_degdaynopre, mo_mpr_global_variables::i1_fasp, mo_global_variables::i1_fastrunoff, mo_mpr_global_variables::i1_froots, mo_mpr_global_variables::i1_fsealed, mo_mpr_global_variables::i1←harsamcoeff, mo_global_variables::i1_infilsoil, mo_global_variables::i1_inter, mo_mpr_global_variables::i1←jarvis_thresh_c1, mo_mpr_global_variables::i1_karstloss, mo_mpr_global_variables::i1_kbaseflow, mo_mpr←_global_variables::i1_kfastflow, mo_mpr_global_variables::i1_kperco, mo_mpr_global_variables::i1_kslowflow, mo_mrm_global_variables::i1_l11_id, mo_mpr_global_variables::i1_maxinter, mo_global_variables::i1_melt, mo←_global_variables::i1_netrad, mo_global_variables::i1_neutrons, mo_global_variables::i1_percol, mo_global←variables::i1_pet, mo_global_variables::i1_pet_calc, mo_global_variables::i1_pet_weights, mo_mpr_global←variables::i1_petlaicfactor, mo_global_variables::i1_pre, mo_global_variables::i1_pre_weights, mo_global←

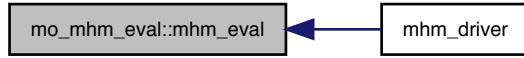
variables::l1_preeffect, mo_mpr_global_variables::l1_prietaryalpha, mo_global_variables::l1_rain, mo_global_variables::l1_runoffseal, mo_global_variables::l1_satstw, mo_mpr_global_variables::l1_sealedthresh, mo_global_variables::l1_sealstw, mo_global_variables::l1_slowrunoff, mo_global_variables::l1_snow, mo_global_variables::l1_snowpack, mo_global_variables::l1_soilmoist, mo_mpr_global_variables::l1_soilmoistexp, mo_mpr_global_variables::l1_soilmoistfc, mo_mpr_global_variables::l1_soilmoistsat, mo_mpr_global_variables::l1_surfresist, mo_global_variables::l1_temp, mo_global_variables::l1_temp_weights, mo_mpr_global_variables::l1_tempthresh, mo_global_variables::l1_throughfall, mo_global_variables::l1_tmax, mo_global_variables::l1_tmin, mo_global_variables::l1_total_runoff, mo_global_variables::l1_unsatstw, mo_mpr_global_variables::l1_unsatthresh, mo_mpr_global_variables::l1_wiltingpoint, mo_global_variables::l1_windspeed, mo_common_mhm_mrm_variables::lcyearid, mo_common_variables::level1, mo_mrm_global_variables::level11, mo_message::message(), mo_mhm::mhm(), mo_mpr_eval::mpr_eval(), mo_mrm_restart::mrm_read_restart_states(), mo_mrm_routing::mrm_routing(), mo_mrm_global_variables::mrm_runoff, mo_mrm_init::mrm_update_param(), mo_mrm_write::mrm_write_output_fluxes(), mo_common_variables::nbasins, mo_global_variables::neutron_integral_afast, mo_write_fluxes_states::newoutputdataset(), mo_common_constants::nodata_dp, mo_mpr_global_variables::nsoilhorizons_mhm, mo_global_variables::nsoilhorizons_sm_input, mo_global_variables::ntimesteps_l1_et, mo_global_variables::ntimesteps_l1_neutrons, mo_global_variables::ntimesteps_l1_sm, mo_common_mhm_mrm_variables::ntstepday, mo_common_mhm_mrm_variables::optimize, mo_global_variables::outputflxstate, mo_mrm_global_variables::outputflxstate_mrm, mo_meteo_forcings::prepare_meteo_forcings_data(), mo_common_variables::processmatrix, mo_global_variables::read_meteo_weights, mo_common_mhm_mrm_variables::read_restart, mo_restart::read_restart_states(), mo_common_mhm_mrm_variables::readper, mo_common_variables::resolutionhydrology, mo_common_mhm_mrm_variables::resolutionrouting, mo_common_mhm_mrm_variables::simper, mo_common_mhm_mrm_variables::timestep, mo_global_variables::timestep_et_input, mo_mpr_global_variables::timestep_lai_input, mo_global_variables::timestep_model_inputs, mo_global_variables::timestep_model_outputs, mo_mrm_global_variables::timestep_model_outputs_mrm, mo_global_variables::timestep_sm_input, mo_init_states::variables_default_init(), mo_mrm_init::variables_default_init_routing(), and mo_common_mhm_mrm_variables::warmingdays.

Referenced by mhm_driver().

Here is the call graph for this function:



Here is the caller graph for this function:



15.34 mo_mhm_read_config Module Reference

Reading of main model configurations.

Functions/Subroutines

- subroutine, public [mhm_read_config](#) (file_namelist, unamelist)
Read main configurations for mHM.

15.34.1 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

15.34.2 Function/Subroutine Documentation

15.34.2.1 mhm_read_config()

```

subroutine, public mo_mhm_read_config::mhm_read_config (
    character(*), intent(in) file_namelist,
    integer, intent(in) unamelist )
  
```

Read main configurations for mHM.

The main configurations in mHM are read from three files:

1. mhm.nml
2. mhm_parameters.nml
3. mhm_outputs.nml

For details please refer to the above mentioned namelist files.

Parameters

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

Authors

Matthias Zink

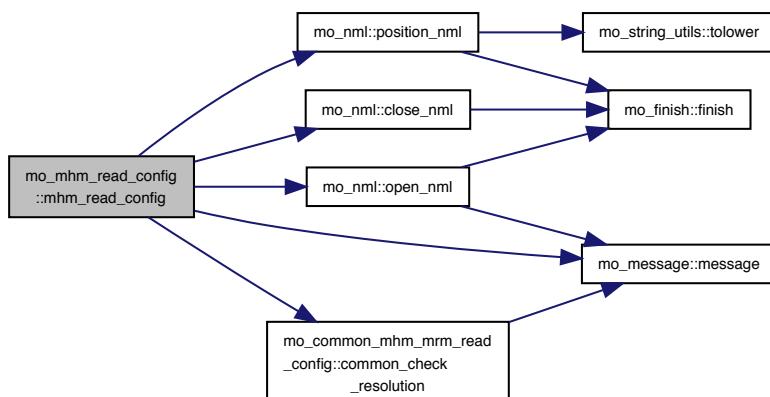
Date

Dec 2012

References mo_global_variables::basin_avg_tws_obs, mo_nml::close_nml(), mo_common_mhm_mrm::read_config::common_check_resolution(), mo_global_variables::dirabsvappressure, mo_global_variables::direvapotranspiration, mo_global_variables::dirmaxtemperature, mo_global_variables::dirmintemperature, mo_global_variables::dirnetradiation, mo_global_variables::dirneutrons, mo_global_variables::dirprecipitation, mo_global_variables::dirreferenceet, mo_global_variables::dirsoil_moisture, mo_global_variables::dirtemperature, mo_global_variables::dirwindspeed, mo_global_variables::evap_coeff, mo_global_variables::fdays_pet, mo_global_variables::fdays_prec, mo_global_variables::fdays_temp, mo_file::file_defoutput, mo_global_variables::filetws, mo_global_variables::fnights_pet, mo_global_variables::fnights_prec, mo_global_variables::fnights_temp, mo_global_variables::inputformat_meteo_forcings, mo_common_constants::maxnbasins, mo_mpr_constants::maxnosoilhorizons, mo_message::message(), mo_common_variables::nbasins, mo_common_constants::nodata_i4, mo_mpr_global_variables::nsoilhorizons_mhm, mo_global_variables::nsoilhorizons_sm_input, mo_nml::open_nml(), mo_common_mhm_mrm_variables::opti_function, mo_common_mhm_mrm_variables::optimize, mo_global_variables::outputflxstate, mo_nml::position_nml(), mo_common_variables::processmatrix, mo_global_variables::read_meteo_weights, mo_global_variables::timestep_et_input, mo_global_variables::timestep_model_inputs, mo_global_variables::timestep_model_outputs, mo_global_variables::timestep_neurons_input, mo_global_variables::timestep_sm_input, and mo_file::udefoutput.

Referenced by mhm_driver().

Here is the call graph for this function:



Here is the caller graph for this function:



15.35 mo_moment Module Reference

Data Types

- interface [absdev](#)
- interface [average](#)
- interface [central_moment](#)
- interface [central_moment_var](#)
- interface [correlation](#)
- interface [covariance](#)
- interface [kurtosis](#)
- interface [mean](#)
- interface [mixed_central_moment](#)
- interface [mixed_central_moment_var](#)
- interface [moment](#)
- interface [skewness](#)
- interface [stddev](#)
- interface [variance](#)

Functions/Subroutines

- real(dp) function [absdev_dp](#) (dat, mask)
- real(sp) function [absdev_sp](#) (dat, mask)
- real(dp) function [average_dp](#) (dat, mask)
- real(sp) function [average_sp](#) (dat, mask)
- real(dp) function [central_moment_dp](#) (x, r, mask)
- real(sp) function [central_moment_sp](#) (x, r, mask)
- real(dp) function [central_moment_var_dp](#) (x, r, mask)
- real(sp) function [central_moment_var_sp](#) (x, r, mask)
- real(dp) function [correlation_dp](#) (x, y, mask)
- real(sp) function [correlation_sp](#) (x, y, mask)
- real(dp) function [covariance_dp](#) (x, y, mask)
- real(sp) function [covariance_sp](#) (x, y, mask)
- real(dp) function [kurtosis_dp](#) (dat, mask)
- real(sp) function [kurtosis_sp](#) (dat, mask)
- real(dp) function [mean_dp](#) (dat, mask)
- real(sp) function [mean_sp](#) (dat, mask)
- real(dp) function [mixed_central_moment_dp](#) (x, y, r, s, mask)
- real(sp) function [mixed_central_moment_sp](#) (x, y, r, s, mask)
- real(dp) function [mixed_central_moment_var_dp](#) (x, y, r, s, mask)

- real(sp) function `mixed_central_moment_var_sp` (x, y, r, s, mask)
- subroutine `moment_dp` (dat, average, variance, skewness, kurtosis, mean, stddev, absdev, mask)
- subroutine `moment_sp` (dat, average, variance, skewness, kurtosis, mean, stddev, absdev, mask)
- real(dp) function `stddev_dp` (dat, mask)
- real(sp) function `stddev_sp` (dat, mask)
- real(dp) function `skewness_dp` (dat, mask)
- real(sp) function `skewness_sp` (dat, mask)
- real(dp) function `variance_dp` (dat, mask)
- real(sp) function `variance_sp` (dat, mask)

15.35.1 Function/Subroutine Documentation

15.35.1.1 `absdev_dp()`

```
real(dp) function mo_moment::absdev_dp (
    real(dp), dimension(:), intent(in) dat,
    logical, dimension(:), intent(in), optional mask ) [private]
```

15.35.1.2 `absdev_sp()`

```
real(sp) function mo_moment::absdev_sp (
    real(sp), dimension(:), intent(in) dat,
    logical, dimension(:), intent(in), optional mask ) [private]
```

15.35.1.3 `average_dp()`

```
real(dp) function mo_moment::average_dp (
    real(dp), dimension(:), intent(in) dat,
    logical, dimension(:), intent(in), optional mask ) [private]
```

15.35.1.4 `average_sp()`

```
real(sp) function mo_moment::average_sp (
    real(sp), dimension(:), intent(in) dat,
    logical, dimension(:), intent(in), optional mask ) [private]
```

15.35.1.5 `central_moment_dp()`

```
real(dp) function mo_moment::central_moment_dp (
    real(dp), dimension(:), intent(in) x,
    integer(i4), intent(in) r,
    logical, dimension(:), intent(in), optional mask ) [private]
```

15.35.1.6 central_moment_sp()

```
real(sp) function mo_moment::central_moment_sp (
    real(sp), dimension(:), intent(in) x,
    integer(i4), intent(in) r,
    logical, dimension(:), intent(in), optional mask ) [private]
```

15.35.1.7 central_moment_var_dp()

```
real(dp) function mo_moment::central_moment_var_dp (
    real(dp), dimension(:), intent(in) x,
    integer(i4), intent(in) r,
    logical, dimension(:), intent(in), optional mask ) [private]
```

15.35.1.8 central_moment_var_sp()

```
real(sp) function mo_moment::central_moment_var_sp (
    real(sp), dimension(:), intent(in) x,
    integer(i4), intent(in) r,
    logical, dimension(:), intent(in), optional mask ) [private]
```

15.35.1.9 correlation_dp()

```
real(dp) function mo_moment::correlation_dp (
    real(dp), dimension(:), intent(in) x,
    real(dp), dimension(:), intent(in) y,
    logical, dimension(:), intent(in), optional mask ) [private]
```

15.35.1.10 correlation_sp()

```
real(sp) function mo_moment::correlation_sp (
    real(sp), dimension(:), intent(in) x,
    real(sp), dimension(:), intent(in) y,
    logical, dimension(:), intent(in), optional mask ) [private]
```

15.35.1.11 covariance_dp()

```
real(dp) function mo_moment::covariance_dp (
    real(dp), dimension(:), intent(in) x,
    real(dp), dimension(:), intent(in) y,
    logical, dimension(:), intent(in), optional mask ) [private]
```

15.35.1.12 covariance_sp()

```
real(sp) function mo_moment::covariance_sp (
```

```
real(sp), dimension(:), intent(in) x,
real(sp), dimension(:), intent(in) y,
logical, dimension(:), intent(in), optional mask ) [private]
```

15.35.1.13 kurtosis_dp()

```
real(dp) function mo_moment::kurtosis_dp (
    real(dp), dimension(:), intent(in) dat,
    logical, dimension(:), intent(in), optional mask ) [private]
```

15.35.1.14 kurtosis_sp()

```
real(sp) function mo_moment::kurtosis_sp (
    real(sp), dimension(:), intent(in) dat,
    logical, dimension(:), intent(in), optional mask ) [private]
```

15.35.1.15 mean_dp()

```
real(dp) function mo_moment::mean_dp (
    real(dp), dimension(:), intent(in) dat,
    logical, dimension(:), intent(in), optional mask ) [private]
```

15.35.1.16 mean_sp()

```
real(sp) function mo_moment::mean_sp (
    real(sp), dimension(:), intent(in) dat,
    logical, dimension(:), intent(in), optional mask ) [private]
```

15.35.1.17 mixed_central_moment_dp()

```
real(dp) function mo_moment::mixed_central_moment_dp (
    real(dp), dimension(:), intent(in) x,
    real(dp), dimension(:), intent(in) y,
    integer(i4), intent(in) r,
    integer(i4), intent(in) s,
    logical, dimension(:), intent(in), optional mask ) [private]
```

15.35.1.18 mixed_central_moment_sp()

```
real(sp) function mo_moment::mixed_central_moment_sp (
    real(sp), dimension(:), intent(in) x,
    real(sp), dimension(:), intent(in) y,
    integer(i4), intent(in) r,
    integer(i4), intent(in) s,
    logical, dimension(:), intent(in), optional mask ) [private]
```

15.35.1.19 mixed_central_moment_var_dp()

```
real(dp) function mo_moment::mixed_central_moment_var_dp (
    real(dp), dimension(:), intent(in) x,
    real(dp), dimension(:), intent(in) y,
    integer(i4), intent(in) r,
    integer(i4), intent(in) s,
    logical, dimension(:), intent(in), optional mask ) [private]
```

15.35.1.20 mixed_central_moment_var_sp()

```
real(sp) function mo_moment::mixed_central_moment_var_sp (
    real(sp), dimension(:), intent(in) x,
    real(sp), dimension(:), intent(in) y,
    integer(i4), intent(in) r,
    integer(i4), intent(in) s,
    logical, dimension(:), intent(in), optional mask ) [private]
```

15.35.1.21 moment_dp()

```
subroutine mo_moment::moment_dp (
    real(dp), dimension(:), intent(in) dat,
    real(dp), intent(out), optional average,
    real(dp), intent(out), optional variance,
    real(dp), intent(out), optional skewness,
    real(dp), intent(out), optional kurtosis,
    real(dp), intent(out), optional mean,
    real(dp), intent(out), optional stddev,
    real(dp), intent(out), optional absdev,
    logical, dimension(:), intent(in), optional mask ) [private]
```

15.35.1.22 moment_sp()

```
subroutine mo_moment::moment_sp (
    real(sp), dimension(:), intent(in) dat,
    real(sp), intent(out), optional average,
    real(sp), intent(out), optional variance,
    real(sp), intent(out), optional skewness,
    real(sp), intent(out), optional kurtosis,
    real(sp), intent(out), optional mean,
    real(sp), intent(out), optional stddev,
    real(sp), intent(out), optional absdev,
    logical, dimension(:), intent(in), optional mask ) [private]
```

15.35.1.23 skewness_dp()

```
real(dp) function mo_moment::skewness_dp (
```

```
real(dp), dimension(:), intent(in) dat,
logical, dimension(:), intent(in), optional mask ) [private]
```

15.35.1.24 skewness_sp()

```
real(sp) function mo_moment::skewness_sp (
    real(sp), dimension(:), intent(in) dat,
    logical, dimension(:), intent(in), optional mask ) [private]
```

15.35.1.25 stddev_dp()

```
real(dp) function mo_moment::stddev_dp (
    real(dp), dimension(:), intent(in) dat,
    logical, dimension(:), intent(in), optional mask ) [private]
```

15.35.1.26 stddev_sp()

```
real(sp) function mo_moment::stddev_sp (
    real(sp), dimension(:), intent(in) dat,
    logical, dimension(:), intent(in), optional mask ) [private]
```

15.35.1.27 variance_dp()

```
real(dp) function mo_moment::variance_dp (
    real(dp), dimension(:), intent(in) dat,
    logical, dimension(:), intent(in), optional mask ) [private]
```

15.35.1.28 variance_sp()

```
real(sp) function mo_moment::variance_sp (
    real(sp), dimension(:), intent(in) dat,
    logical, dimension(:), intent(in), optional mask ) [private]
```

15.36 mo_mpr_constants Module Reference

Provides MPR specific constants.

Variables

- integer(i4), parameter, public **nlcover_class** = 3_i4
- integer(i4), parameter, public **maxgeounit** = 25_i4
- integer(i4), parameter, public **maxnosoilhorizons** = 10_i4
- real(dp), parameter, public **p2_initstatefluxes** = 15.00_dp
- real(dp), parameter, public **p3_initstatefluxes** = 10.00_dp

- real(dp), parameter, public `p4_initstatefluxes` = 75.00_dp
- real(dp), parameter, public `p5_initstatefluxes` = 1500.00_dp
- real(dp), parameter, public `c1_initstatesm` = 0.25_dp
- real(dp), parameter, public `bulkdens_orgmatter` = 0.224_dp
- real(dp), parameter, public `field_cap_c1` = -0.60_dp
- real(dp), parameter, public `field_cap_c2` = 2.0_dp
- real(dp), parameter, public `vgenuchten_sandtresh` = 66.5_dp
- real(dp), parameter, public `vgenuchtenn_c1` = 1.392_dp
- real(dp), parameter, public `vgenuchtenn_c2` = 0.418_dp
- real(dp), parameter, public `vgenuchtenn_c3` = -0.024_dp
- real(dp), parameter, public `vgenuchtenn_c4` = 1.212_dp
- real(dp), parameter, public `vgenuchtenn_c5` = -0.704_dp
- real(dp), parameter, public `vgenuchtenn_c6` = -0.648_dp
- real(dp), parameter, public `vgenuchtenn_c7` = 0.023_dp
- real(dp), parameter, public `vgenuchtenn_c8` = 0.044_dp
- real(dp), parameter, public `vgenuchtenn_c9` = 3.168_dp
- real(dp), parameter, public `vgenuchtenn_c10` = -2.562_dp
- real(dp), parameter, public `vgenuchtenn_c11` = 7.0E-9_dp
- real(dp), parameter, public `vgenuchtenn_c12` = 4.004_dp
- real(dp), parameter, public `vgenuchtenn_c13` = 3.750_dp
- real(dp), parameter, public `vgenuchtenn_c14` = -0.016_dp
- real(dp), parameter, public `vgenuchtenn_c15` = -4.197_dp
- real(dp), parameter, public `vgenuchtenn_c16` = 0.013_dp
- real(dp), parameter, public `vgenuchtenn_c17` = 0.076_dp
- real(dp), parameter, public `vgenuchtenn_c18` = 0.276_dp
- real(dp), parameter, public `ks_c` = 10.0_dp
- real(dp), parameter, public `pwp_c` = 1.0_dp
- real(dp), parameter, public `pwp_matpot_theta` = 15000.0_dp
- real(dp), parameter, public `windmeasheight` = 10.0_dp

assumed meteorol. measurement height for estimation of aeroResist and surfResist

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

von karman constant

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

LAI factor for bulk surface resistance formulation.

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

LAI offset for bulk surface resistance formulation.

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

maximum bulk surface resistance

15.36.1 Detailed Description

Provides MPR specific constants.

Provides MPR specific constants such as flood plain elevation.

Authors

Matthias Cuntz

Date

Nov 2011

15.36.2 Variable Documentation

15.36.2.1 bulkdens_orgmatter

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

Referenced by mo_mpr_soilmoist::mpr_sm().

15.36.2.2 c1_initstatesm

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

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

15.36.2.3 field_cap_c1

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

Referenced by mo_mpr_soilmoist::field_cap().

15.36.2.4 field_cap_c2

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

Referenced by mo_mpr_soilmoist::field_cap().

15.36.2.5 karman

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

von Karman constant

Referenced by mo_multi_param_reg::aerodynamical_resistance().

15.36.2.6 ks_c

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

Referenced by mo_mpr_soilmoist::hydro_cond().

15.36.2.7 lai_factor_surfresi

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

LAI factor for bulk surface resistance formulation.

Referenced by mo_mpr_pet::bulksurface_resistance().

15.36.2.8 lai_offset_surfresi

```
real(dp), parameter, public mo_mpr_constants::lai_offset_surfresi = 1.2_dp
LAI offset for bulk surface resistance formulation.
```

Referenced by `mo_mpr_pet::bulksurface_resistance()`.

15.36.2.9 max_surfresist

```
real(dp), parameter, public mo_mpr_constants::max_surfresist = 250.0_dp
maximum bulk surface resistance
```

Referenced by `mo_mpr_pet::bulksurface_resistance()`.

15.36.2.10 maxgeounit

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

Referenced by `mo_mpr_read_config::mpr_read_config()`.

15.36.2.11 maxnosoilhorizons

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

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

15.36.2.12 nlcover_class

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

Referenced by `mo_soil_database::read_soil_lut()`.

15.36.2.13 p2_initstatefluxes

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

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

15.36.2.14 p3_initstatefluxes

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

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

15.36.2.15 p4_initstatefluxes

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

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

15.36.2.16 p5_initstatefluxes

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

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

15.36.2.17 pwp_c

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

Referenced by `mo_mpr_soilmoist::pwp()`.

15.36.2.18 pwp_matpot_theta

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

Referenced by `mo_mpr_soilmoist::pwp()`.

15.36.2.19 vgenuchten_sandtresh

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

Referenced by `mo_mpr_soilmoist::genuchten()`.

15.36.2.20 vgenuchtenn_c1

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

Referenced by `mo_mpr_soilmoist::genuchten()`.

15.36.2.21 vgenuchtenn_c10

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

Referenced by `mo_mpr_soilmoist::genuchten()`.

15.36.2.22 vgenuchtenn_c11

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

Referenced by `mo_mpr_soilmoist::genuchten()`.

15.36.2.23 vgenuchten_c12

```
real(dp), parameter, public mo_mpr_constants::vgenuchten_c12 = 4.004_dp
```

Referenced by mo_mpr_soilmoist::genuchten().

15.36.2.24 vgenuchten_c13

```
real(dp), parameter, public mo_mpr_constants::vgenuchten_c13 = 3.750_dp
```

Referenced by mo_mpr_soilmoist::genuchten().

15.36.2.25 vgenuchten_c14

```
real(dp), parameter, public mo_mpr_constants::vgenuchten_c14 = -0.016_dp
```

Referenced by mo_mpr_soilmoist::genuchten().

15.36.2.26 vgenuchten_c15

```
real(dp), parameter, public mo_mpr_constants::vgenuchten_c15 = -4.197_dp
```

Referenced by mo_mpr_soilmoist::genuchten().

15.36.2.27 vgenuchten_c16

```
real(dp), parameter, public mo_mpr_constants::vgenuchten_c16 = 0.013_dp
```

Referenced by mo_mpr_soilmoist::genuchten().

15.36.2.28 vgenuchten_c17

```
real(dp), parameter, public mo_mpr_constants::vgenuchten_c17 = 0.076_dp
```

Referenced by mo_mpr_soilmoist::genuchten().

15.36.2.29 vgenuchten_c18

```
real(dp), parameter, public mo_mpr_constants::vgenuchten_c18 = 0.276_dp
```

Referenced by mo_mpr_soilmoist::genuchten().

15.36.2.30 vgenuchtenn_c2

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

Referenced by mo_mpr_soilmoist::genuchten().

15.36.2.31 vgenuchtenn_c3

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

Referenced by mo_mpr_soilmoist::genuchten().

15.36.2.32 vgenuchtenn_c4

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

Referenced by mo_mpr_soilmoist::genuchten().

15.36.2.33 vgenuchtenn_c5

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

Referenced by mo_mpr_soilmoist::genuchten().

15.36.2.34 vgenuchtenn_c6

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

Referenced by mo_mpr_soilmoist::genuchten().

15.36.2.35 vgenuchtenn_c7

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

Referenced by mo_mpr_soilmoist::genuchten().

15.36.2.36 vgenuchtenn_c8

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

Referenced by mo_mpr_soilmoist::genuchten().

15.36.2.37 vgenuchtenn_c9

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

Referenced by mo_mpr_soilmoist::genuchten().

15.36.2.38 windmeasheight

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

assumed meteorol. measurement height for estimation of aeroResist and surfResist

Referenced by mo_multi_param_reg::aerodynamical_resistance().

15.37 mo_mpr_eval Module Reference

Runs MPR and writes to global effective parameters.

Functions/Subroutines

- subroutine, public mpr_eval (parameterset)
Runs MPR and writes to global effective parameters.

15.37.1 Detailed Description

Runs MPR and writes to global effective parameters.

Runs MPR and writes to global effective parameters

Authors

Robert Schweiß

Date

Feb 2018

15.37.2 Function/Subroutine Documentation

15.37.2.1 mpr_eval()

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

Runs MPR and writes to global effective parameters.

Runs MPR and writes to global effective parameters

Parameters

in	real(dp), dimension(:), optional :: parameterset	a set of global parameter (gamma) to run mHM, DIMENSION [no. of global_Parameters]
----	--	--

Authors

Juliane Mai, Rohini Kumar

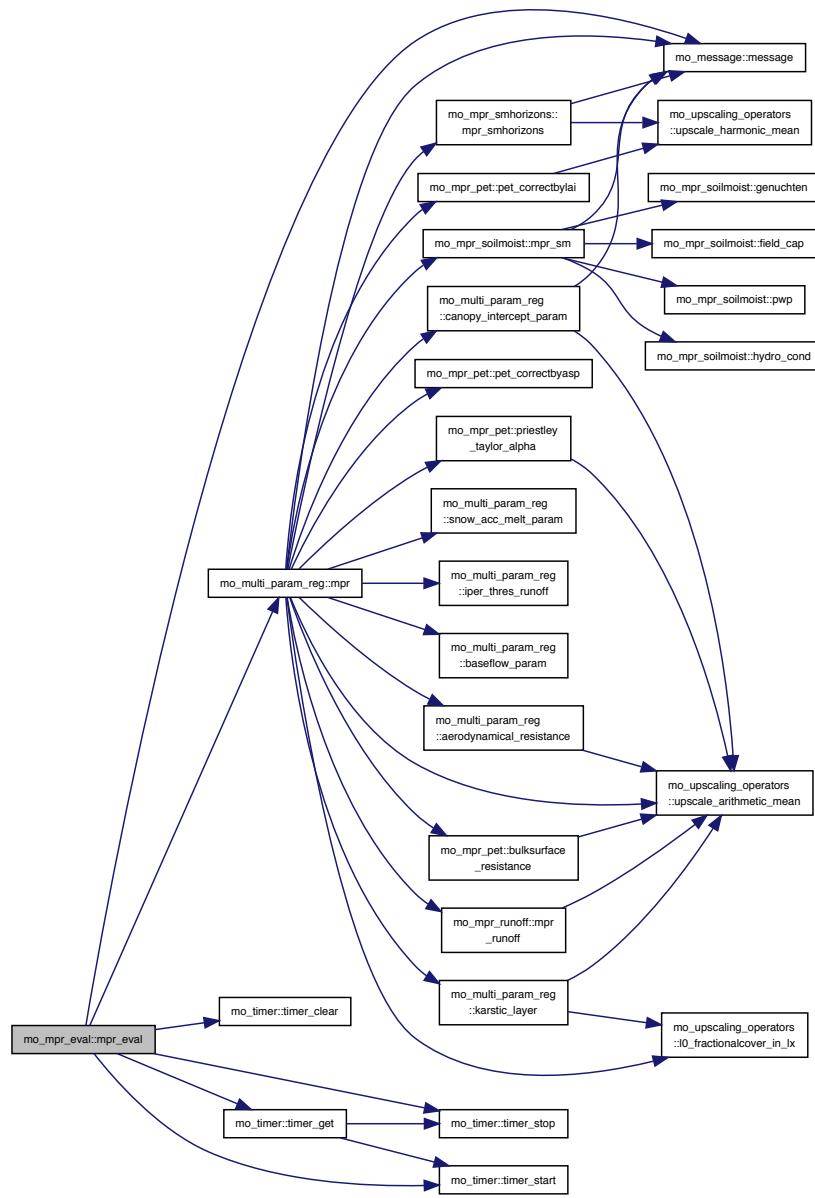
Date

Feb 2013

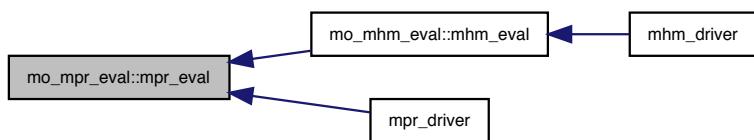
References mo_mpr_global_variables::l0_asp, mo_common_variables::l0_basin, mo_mpr_global_variables::l0_geounit, mo_mpr_global_variables::l0_gridded_lai, mo_common_variables::l0_l1_remap, mo_common_variables::l0_lcover, mo_mpr_global_variables::l0_slope_emp, mo_mpr_global_variables::l0_soilid, mo_mpr_global_variables::l1_aeroresist, mo_mpr_global_variables::l1_alpha, mo_mpr_global_variables::l1_degdayinc, mo_mpr_global_variables::l1_degdaymax, mo_mpr_global_variables::l1_degdaynopre, mo_mpr_global_variables::l1_fasp, mo_mpr_global_variables::l1_froots, mo_mpr_global_variables::l1_fsealed, mo_mpr_global_variables::l1_harsamcoeff, mo_mpr_global_variables::l1_jarvis_thresh_c1, mo_mpr_global_variables::l1_karstloss, mo_mpr_global_variables::l1_kbaseflow, mo_mpr_global_variables::l1_kfastflow, mo_mpr_global_variables::l1_kperco, mo_mpr_global_variables::l1_kslowflow, mo_mpr_global_variables::l1_maxinter, mo_mpr_global_variables::l1_petlaicfactor, mo_mpr_global_variables::l1_prietayalpha, mo_mpr_global_variables::l1_sealedthresh, mo_mpr_global_variables::l1_soilmoistexp, mo_mpr_global_variables::l1_soilmoistfc, mo_mpr_global_variables::l1_soilmoistsat, mo_mpr_global_variables::l1_surfresist, mo_mpr_global_variables::l1_tempthresh, mo_mpr_global_variables::l1_unsatthresh, mo_mpr_global_variables::l1_wiltingpoint, mo_common_variables::level0, mo_common_variables::level1, mo_message::message(), mo_multi_param_reg::mpr(), mo_common_variables::nbasins, mo_timer::timer_clear(), mo_timer::timer_get(), mo_timer::timer_start(), and mo_timer::timer_stop().

Referenced by mo_mhm_eval::mhm_eval(), and mpr_driver().

Here is the call graph for this function:



Here is the caller graph for this function:



15.38 mo_mpr_file Module Reference

Provides file names and units for mRM.

Variables

- character(len=*), parameter **version** = '0.1'
Current mHM model version.
- character(len=*), parameter **version_date** = 'Jun 2018'
Time of current mHM model version release.
- character(len=*), parameter **file_main** = 'mpr_driver.f90'
Driver file.
- character(len=*), parameter **file_namelist_mpr** = 'mpr.nml'
Namelist file name.
- integer, parameter **unamelist_mpr** = 80
Unit for namelist.
- character(len=*), parameter **file_namelist_mpr_param** = 'mpr_parameter.nml'
Parameter namelists file name.
- integer, parameter **unamelist_mpr_param** = 31
Unit for namelist.
- character(len=*), parameter **file_soil_database** = 'soil_classdefinition.txt'
Soil database file (iFlag_soilDB = 0) = classical mHM format.
- character(len=*), parameter **file_soil_database_1** = 'soil_classdefinition_iFlag_soilDB_1.txt'
Soil database file (iFlag_soilDB = 1)
- integer, parameter **usoil_database** = 52
Unit for soil data base.
- character(len=*), parameter **file_slope** = 'slope.asc'
slope input data file
- integer, parameter **uslope** = 54
Unit for slope input data file.
- character(len=*), parameter **file_aspect** = 'aspect.asc'
aspect input data file
- integer, parameter **uaspect** = 55
Unit for aspect input data file.
- character(len=*), parameter **file_hydrogeoclass** = 'geology_class.asc'
hydrogeological classes input data file
- integer, parameter **uhydrogeoclass** = 58
Unit for hydrogeological classes input data file.
- character(len=*), parameter **file_soilclass** = 'soil_class.asc'
soil classes input data file
- integer, parameter **usoilclass** = 59
Unit for soil classes input data file.
- character(len=*), parameter **file_laiclass** = 'LAI_class.asc'
LAI classes input data file.
- integer, parameter **ulaiclass** = 60
Unit for LAI input data file.
- character(len=*), parameter **file_geolut** = 'geology_classdefinition.txt'
geological formation lookup table file
- integer, parameter **ugeolut** = 64
Unit for geological formation lookup table file.

- character(len=*), parameter **file_lailut** = 'LAI_classdefinition.txt'
LAI classes lookup table file.
- integer, parameter **ulailut** = 65
Unit for LAI classes lookup table file.
- character(len=*), parameter **file_meteo_header** = 'header.txt'
Input nCols and nRows of binary meteo and LAI files are in header file.
- integer, parameter **umeteo_header** = 50
Unit for meteo header file.
- character(len=*), parameter **file_meteo_binary_end** = '.bin'
File ending of meteo files.
- integer, parameter **umeteo** = 51
Unit for meteo files.

15.38.1 Detailed Description

Provides file names and units for mRM.

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

Authors

Matthias Cuntz, Stephan Thober

Date

Aug 2015

15.38.2 Variable Documentation

15.38.2.1 file_aspect

character(len = *), parameter mo_mpr_file::file_aspect = 'aspect.asc'
 aspect input data file

Referenced by `mo_read_wrapper::read_data()`.

15.38.2.2 file_geolut

character(len = *), parameter mo_mpr_file::file_geolut = 'geology_classdefinition.txt'
 geological formation lookup table file

Referenced by `mo_read_wrapper::read_data()`.

15.38.2.3 file_hydrogeoclass

character(len = *), parameter mo_mpr_file::file_hydrogeoclass = 'geology_class.asc'
 hydrogeological classes input data file

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

15.38.2.4 file_laiclass

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

LAI classes input data file.

Referenced by mo_read_wrapper::read_data().

15.38.2.5 file_lailut

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

LAI classes lookup table file.

Referenced by mo_read_wrapper::read_data().

15.38.2.6 file_main

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

Driver file.

15.38.2.7 file_meteo_binary_end

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

File ending of meteo files.

15.38.2.8 file_meteo_header

```
character(len = *), parameter mo_mpr_file::file_meteo_header = 'header.txt'
```

Input nCols and nRows of binary meteo and LAI files are in header file.

Referenced by mo_startup::i2_variable_init().

15.38.2.9 file_namelist_mpr

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

Namelist file name.

15.38.2.10 file_namelist_mpr_param

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

Parameter namelists file name.

Referenced by mpr_driver().

15.38.2.11 file_slope

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

slope input data file

Referenced by mo_read_wrapper::read_data().

15.38.2.12 file_soil_database

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

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

Referenced by mo_read_wrapper::read_data().

15.38.2.13 file_soil_database_1

```
character(len = *), parameter mo_mpr_file::file_soil_database_1 = 'soil_classdefinition_iFlag←  
_soilDB_1.txt'
```

Soil database file (iFlag_soilDB = 1)

Referenced by mo_read_wrapper::read_data().

15.38.2.14 file_soilclass

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

soil classes input data file

Referenced by mo_read_wrapper::read_data().

15.38.2.15 uaspect

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

Unit for aspect input data file.

Referenced by mo_read_wrapper::read_data().

15.38.2.16 ugeolut

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

Unit for geological formation lookup table file.

Referenced by mo_read_wrapper::read_data().

15.38.2.17 uhydrogeoclass

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

Unit for hydrogeological classes input data file.

Referenced by mo_read_wrapper::read_data().

15.38.2.18 ulaiclass

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

Unit for LAI input data file.

Referenced by mo_read_wrapper::read_data().

15.38.2.19 ulailut

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

Unit for LAI classes lookup table file.

Referenced by mo_read_wrapper::read_data().

15.38.2.20 umeteo

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

Unit for meteo files.

15.38.2.21 umeteo_header

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

Unit for meteo header file.

Referenced by mo_startup::l2_variable_init().

15.38.2.22 unamelist_mpr

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

Unit for namelist.

15.38.2.23 unamelist_mpr_param

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

Unit for namelist.

Referenced by mpr_driver().

15.38.2.24 uslope

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

Unit for slope input data file.

Referenced by `mo_read_wrapper::read_data()`.

15.38.2.25 usoil_database

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

Unit for soil data base.

Referenced by `mo_soil_database::read_soil_lut()`.

15.38.2.26 usoilclass

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

Unit for soil classes input data file.

Referenced by `mo_read_wrapper::read_data()`.

15.38.2.27 version

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

Current mHM model version.

15.38.2.28 version_date

```
character(len = *), parameter mo_mpr_file::version_date = 'Jun 2018'
```

Time of current mHM model version release.

15.39 mo_mpr_global_variables Module Reference

Global variables for mpr only.

Data Types

- type [soiltype](#)

Variables

- `real(dp), public c2tstu`

- real(dp), public `tillagedepth`
- integer(i4), public `nsoiltypes`
- integer(i4), public `iflag_soildb`
- integer(i4), public `nsoilhorizons_mhm`
- real(dp), dimension(:), allocatable, public `horizondepth_mhm`
- type(`soiltype`), public `soildb`
- integer(i4), public `ngeounits`
- integer(i4), dimension(:), allocatable, public `geounitlist`
- integer(i4), dimension(:), allocatable, public `geounitkar`
- character(256), public `inputformat_gridded_lai`
- integer(i4), public `timestep_lai_input`
- integer(i4), public `nlaiclass`
- integer(i4), public `nlai`
- integer(i4), dimension(:), allocatable, public `laiunitlist`
- real(dp), dimension(:, :, :), allocatable, public `lailut`
- type(`period`), dimension(:, :), allocatable, public `laiper`
- real(dp), public `fracsealed_cityarea`
- real(dp), dimension(:, :), allocatable, public `l0_slope_emp`
- real(dp), dimension(:, :, :), allocatable, public `l0_gridded_lai`
- real(dp), dimension(:, :), allocatable, public `l0_slope`
- real(dp), dimension(:, :), allocatable, public `l0_asp`
- integer(i4), dimension(:, :, :), allocatable, public `l0_soilid`
- integer(i4), dimension(:, :), allocatable, public `l0_geounit`
- character(256), dimension(:, :), allocatable, public `dirgridded_lai`
- real(dp), dimension(:, :, :, :), allocatable, public `l1_sealed`
- real(dp), dimension(:, :, :, :), allocatable, public `l1_alpha`
- real(dp), dimension(:, :, :, :), allocatable, public `l1_deggdayinc`
- real(dp), dimension(:, :, :, :), allocatable, public `l1_deggdaymax`
- real(dp), dimension(:, :, :, :), allocatable, public `l1_deggdaynopr`
- real(dp), dimension(:, :, :, :), allocatable, public `l1_deggday`
- real(dp), dimension(:, :, :, :), allocatable, public `l1_karstloss`
- real(dp), dimension(:, :, :, :), allocatable, public `l1_fasp`
- real(dp), dimension(:, :, :, :), allocatable, public `l1_petlaicorfactor`
- real(dp), dimension(:, :, :, :), allocatable, public `l1_harsamcoeff`
- real(dp), dimension(:, :, :, :), allocatable, public `l1_prietaryalpha`
- real(dp), dimension(:, :, :, :), allocatable, public `l1_aeroresist`
- real(dp), dimension(:, :, :, :), allocatable, public `l1_surfresist`
- real(dp), dimension(:, :, :, :), allocatable, public `l1_froots`
- real(dp), dimension(:, :, :, :), allocatable, public `l1_maxinter`
- real(dp), dimension(:, :, :, :), allocatable, public `l1_kfastflow`
- real(dp), dimension(:, :, :, :), allocatable, public `l1_kslowflow`
- real(dp), dimension(:, :, :, :), allocatable, public `l1_kbaseflow`
- real(dp), dimension(:, :, :, :), allocatable, public `l1_kperco`
- real(dp), dimension(:, :, :, :), allocatable, public `l1_soilmoistfc`
- real(dp), dimension(:, :, :, :), allocatable, public `l1_soilmoistsat`
- real(dp), dimension(:, :, :, :), allocatable, public `l1_soilmoistexp`
- real(dp), dimension(:, :, :, :), allocatable, public `l1_jarvis_thresh_c1`
- real(dp), dimension(:, :, :, :), allocatable, public `l1_tempthresh`
- real(dp), dimension(:, :, :, :), allocatable, public `l1_unsatthresh`
- real(dp), dimension(:, :, :, :), allocatable, public `l1_sealedthresh`
- real(dp), dimension(:, :, :, :), allocatable, public `l1_wiltingpoint`

15.39.1 Detailed Description

Global variables for mpr only.

TODO: add description

Authors

Robert Scheweppe

Date

Dec 2017

15.39.2 Variable Documentation

15.39.2.1 c2tstu

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

Referenced by mo_startup::constants_init(), mo_multi_param_reg::karstic_layer(), mo_multi_param_reg::mpr(), and mpr_driver().

15.39.2.2 dirgridded_lai

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

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

15.39.2.3 fracsealed_cityarea

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

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

15.39.2.4 geounitkar

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

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

15.39.2.5 geounitlist

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

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

15.39.2.6 horizontodepth_mhm

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

Referenced by mo_soil_database::generate_soil_database(), mo_mhm_eval::mhm_eval(), mo_multi_param_reg::mpr(), mo_mpr_read_config::mpr_read_config(), mo_soil_database::read_soil_lut(), mo_init_states::variables_alloc(), and mo_init_states::variables_default_init().

15.39.2.7 iflag_soildb

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

Referenced by mo_soil_database::generate_soil_database(), mo_mpr_startup::l0_check_input(), mo_mpr_startup::l0_variable_init(), mo_multi_param_reg::mpr(), mo_mpr_read_config::mpr_read_config(), mo_mpr_soilmoist::mpr_sm(), mo_read_wrapper::read_data(), and mo_soil_database::read_soil_lut().

15.39.2.8 inputformat_gridded_lai

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

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

15.39.2.9 l0_asp

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

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

15.39.2.10 l0_geounit

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

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

15.39.2.11 l0_gridded_lai

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

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

15.39.2.12 l0_slope

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

Referenced by `mo_mpr_startup::l0_check_input()`, `mo_mpr_startup::l0_variable_init()`, and `mo_read_wrapper::read_data()`.

15.39.2.13 l0_slope_emp

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

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

15.39.2.14 l0_soilid

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

Referenced by `mo_mpr_startup::l0_check_input()`, `mo_mpr_startup::l0_variable_init()`, `mo_mpr_eval::mpr_eval()`, and `mo_read_wrapper::read_data()`.

15.39.2.15 l1_aeroresist

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

Referenced by `mo_mpr_startup::init_eff_params()`, `mo_mhm_eval::mhm_eval()`, `mo_mpr_eval::mpr_eval()`, `mo_restart::read_restart_states()`, `mo_init_states::variables_default_init()`, and `mo_mpr_restart::write_eff_params()`.

15.39.2.16 l1_alpha

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

Referenced by `mo_mpr_startup::init_eff_params()`, `mo_mhm_eval::mhm_eval()`, `mo_mpr_eval::mpr_eval()`, `mo_restart::read_restart_states()`, `mo_init_states::variables_default_init()`, and `mo_mpr_restart::write_eff_params()`.

15.39.2.17 l1_deggday

```
real(dp), dimension(:, :, :, :, :, :), allocatable, public mo_mpr_global_variables::l1_deggday
```

Referenced by `mo_mpr_startup::init_eff_params()`, `mo_mhm_eval::mhm_eval()`, `mo_restart::read_restart_states()`, `mo_init_states::variables_default_init()`, and `mo_mpr_restart::write_eff_params()`.

15.39.2.18 l1_deggdayinc

```
real(dp), dimension(:, :, :, :, :, :, :), allocatable, public mo_mpr_global_variables::l1_deggdayinc
```

Referenced by `mo_mpr_startup::init_eff_params()`, `mo_mhm_eval::mhm_eval()`, `mo_mpr_eval::mpr_eval()`, `mo_restart::read_restart_states()`, `mo_init_states::variables_default_init()`, and `mo_mpr_restart::write_eff_params()`.

15.39.2.19 l1_degdaymax

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

Referenced by mo_mpr_startup::init_eff_params(), mo_mhm_eval::mhm_eval(), mo_mpr_eval::mpr_eval(), mo_restart::read_restart_states(), mo_init_states::variables_default_init(), and mo_mpr_restart::write_eff_params().

15.39.2.20 l1_degdaynopre

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

Referenced by mo_mpr_startup::init_eff_params(), mo_mhm_eval::mhm_eval(), mo_mpr_eval::mpr_eval(), mo_restart::read_restart_states(), mo_init_states::variables_default_init(), and mo_mpr_restart::write_eff_params().

15.39.2.21 l1_fasp

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

Referenced by mo_mpr_startup::init_eff_params(), mo_mhm_eval::mhm_eval(), mo_mpr_eval::mpr_eval(), mo_restart::read_restart_states(), mo_init_states::variables_default_init(), and mo_mpr_restart::write_eff_params().

15.39.2.22 l1_froots

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

Referenced by mo_mpr_startup::init_eff_params(), mo_mhm_eval::mhm_eval(), mo_mpr_eval::mpr_eval(), mo_restart::read_restart_states(), mo_init_states::variables_default_init(), and mo_mpr_restart::write_eff_params().

15.39.2.23 l1_fsealed

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

Referenced by mo_mpr_startup::init_eff_params(), mo_mhm_eval::mhm_eval(), mo_mpr_eval::mpr_eval(), mo_restart::read_restart_states(), mo_init_states::variables_default_init(), and mo_mpr_restart::write_eff_params().

15.39.2.24 l1_harsamcoeff

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

Referenced by mo_mpr_startup::init_eff_params(), mo_mhm_eval::mhm_eval(), mo_mpr_eval::mpr_eval(), mo_restart::read_restart_states(), mo_init_states::variables_default_init(), and mo_mpr_restart::write_eff_params().

15.39.2.25 l1_jarvis_thresh_c1

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

Referenced by mo_mpr_startup::init_eff_params(), mo_mhm_eval::mhm_eval(), mo_mpr_eval::mpr_eval(), mo_restart::read_restart_states(), mo_init_states::variables_default_init(), and mo_mpr_restart::write_eff_params().

15.39.2.26 l1_karstloss

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

Referenced by mo_mpr_startup::init_eff_params(), mo_mhm_eval::mhm_eval(), mo_mpr_eval::mpr_eval(), mo_restart::read_restart_states(), mo_init_states::variables_default_init(), and mo_mpr_restart::write_eff_params().

15.39.2.27 l1_kbaseflow

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

Referenced by mo_mpr_startup::init_eff_params(), mo_mhm_eval::mhm_eval(), mo_mpr_eval::mpr_eval(), mo_restart::read_restart_states(), mo_init_states::variables_default_init(), and mo_mpr_restart::write_eff_params().

15.39.2.28 l1_kfastflow

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

Referenced by mo_mpr_startup::init_eff_params(), mo_mhm_eval::mhm_eval(), mo_mpr_eval::mpr_eval(), mo_restart::read_restart_states(), mo_init_states::variables_default_init(), and mo_mpr_restart::write_eff_params().

15.39.2.29 l1_kperco

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

Referenced by mo_mpr_startup::init_eff_params(), mo_mhm_eval::mhm_eval(), mo_mpr_eval::mpr_eval(), mo_restart::read_restart_states(), mo_init_states::variables_default_init(), and mo_mpr_restart::write_eff_params().

15.39.2.30 l1_kslowflow

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

Referenced by mo_mpr_startup::init_eff_params(), mo_mhm_eval::mhm_eval(), mo_mpr_eval::mpr_eval(), mo_restart::read_restart_states(), mo_init_states::variables_default_init(), and mo_mpr_restart::write_eff_params().

15.39.2.31 l1_maxinter

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

Referenced by mo_mpr_startup::init_eff_params(), mo_mhm_eval::mhm_eval(), mo_mpr_eval::mpr_eval(), mo_restart::read_restart_states(), mo_init_states::variables_default_init(), and mo_mpr_restart::write_eff_params().

15.39.2.32 l1_petlaicfactor

```
real(dp), dimension(:, :, :, allocatable, public mo_mpr_global_variables::l1_petlaicfactor
```

Referenced by mo_mpr_startup::init_eff_params(), mo_mhm_eval::mhm_eval(), mo_mpr_eval::mpr_eval(), mo_restart::read_restart_states(), mo_init_states::variables_default_init(), and mo_mpr_restart::write_eff_params().

15.39.2.33 l1_prietaryalpha

```
real(dp), dimension(:, :, :, :), allocatable, public mo_mpr_global_variables::l1_prietaryalpha
```

Referenced by mo_mpr_startup::init_eff_params(), mo_mhm_eval::mhm_eval(), mo_mpr_eval::mpr_eval(), mo_restart::read_restart_states(), mo_init_states::variables_default_init(), and mo_mpr_restart::write_eff_params().

15.39.2.34 l1_sealedthresh

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

Referenced by mo_mpr_startup::init_eff_params(), mo_mhm_eval::mhm_eval(), mo_mpr_eval::mpr_eval(), mo_restart::read_restart_states(), mo_init_states::variables_default_init(), and mo_mpr_restart::write_eff_params().

15.39.2.35 l1_soilmoistexp

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

Referenced by mo_mpr_startup::init_eff_params(), mo_mhm_eval::mhm_eval(), mo_mpr_eval::mpr_eval(), mo_restart::read_restart_states(), mo_init_states::variables_default_init(), and mo_mpr_restart::write_eff_params().

15.39.2.36 l1_soilmoistfc

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

Referenced by mo_mpr_startup::init_eff_params(), mo_mhm_eval::mhm_eval(), mo_mpr_eval::mpr_eval(), mo_restart::read_restart_states(), mo_init_states::variables_default_init(), and mo_mpr_restart::write_eff_params().

15.39.2.37 l1_soilmoistsat

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

Referenced by mo_mpr_startup::init_eff_params(), mo_mhm_eval::mhm_eval(), mo_mpr_eval::mpr_eval(), mo_restart::read_restart_states(), mo_init_states::variables_default_init(), and mo_mpr_restart::write_eff_params().

15.39.2.38 l1_surfresist

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

Referenced by mo_mpr_startup::init_eff_params(), mo_mhm_eval::mhm_eval(), mo_mpr_eval::mpr_eval(), mo_restart::read_restart_states(), mo_init_states::variables_default_init(), and mo_mpr_restart::write_eff_params().

15.39.2.39 l1_tempthresh

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

Referenced by mo_mpr_startup::init_eff_params(), mo_mhm_eval::mhm_eval(), mo_mpr_eval::mpr_eval(), mo_restart::read_restart_states(), mo_init_states::variables_default_init(), and mo_mpr_restart::write_eff_params().

15.39.2.40 l1_unsatthresh

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

Referenced by mo_mpr_startup::init_eff_params(), mo_mhm_eval::mhm_eval(), mo_mpr_eval::mpr_eval(), mo_restart::read_restart_states(), mo_init_states::variables_default_init(), and mo_mpr_restart::write_eff_params().

15.39.2.41 l1_wiltingpoint

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

Referenced by mo_mpr_startup::init_eff_params(), mo_mhm_eval::mhm_eval(), mo_mpr_eval::mpr_eval(), mo_restart::read_restart_states(), mo_init_states::variables_default_init(), and mo_mpr_restart::write_eff_params().

15.39.2.42 lailut

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

Referenced by mo_read_wrapper::read_data().

15.39.2.43 laiper

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

15.39.2.44 laiunitlist

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

Referenced by mo_read_wrapper::read_data().

15.39.2.45 ngeounits

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

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

15.39.2.46 nlai

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

Referenced by mo_mpr_startup::init_eff_params(), mo_prepare_gridded_lai::prepare_gridded_daily_lai_data(), mo_prepare_gridded_lai::prepare_gridded_mean_monthly_lai_data(), mo_read_wrapper::read_data(), mo_restart::read_restart_states(), mo_mpr_restart::write_mpr_restart_files(), and mo_restart::write_restart_files().

15.39.2.47 nlaiclass

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

Referenced by mo_read_wrapper::read_data().

15.39.2.48 nsoilhorizons_mhm

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

Referenced by mo_soil_database::generate_soil_database(), mo_mpr_startup::init_eff_params(), mo_mpr_startup::l0_check_input(), mo_mpr_startup::l0_variable_init(), mo_mhm_eval::mhm_eval(), mo_mhm_read_config::mhm_read_config(), mo_multi_param_reg::mpr(), mo_mpr_read_config::mpr_read_config(), mo_write_fluxes_states::newoutputdataset(), mo_read_wrapper::read_data(), mo_restart::read_restart_states(), mo_soil_database::read_soil_lut(), mo_write_fluxes_states::updatedataset(), mo_init_states::variables_alloc(), mo_init_states::variables_default_init(), mo_mpr_restart::write_mpr_restart_files(), and mo_restart::write_restart_files().

15.39.2.49 nsoiltypes

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

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

15.39.2.50 soildb

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

Referenced by mo_soil_database::generate_soil_database(), mo_mpr_startup::l0_variable_init(), mo_multi_param_reg::mpr(), mo_read_wrapper::read_data(), and mo_soil_database::read_soil_lut().

15.39.2.51 tillagedepth

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

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

15.39.2.52 timestep_lai_input

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

Referenced by mo_mpr_startup::l0_check_input(), mo_mhm_eval::mhm_eval(), mo_mpr_read_config::mpr_read_config(), mo_prepare_gridded_lai::prepare_gridded_daily_lai_data(), and mo_read_wrapper::read_data().

15.40 mo_mpr_pet Module Reference

TODO: add description.

Functions/Subroutines

- subroutine, public `pet_correctbylai` (param, nodata, LCOVER0, LAI0, mask0, cell_id0, upp_row_L1, low_row_L1, lef_col_L1, rig_col_L1, nL0_in_L1, L1_petLAlcorFactor)
estimate PET correction factor based on LAI at L1
- subroutine, public `pet_correctbyasp` (Id0, latitude_I0, Asp0, param, nodata, fAsp0)
correction of PET
- subroutine, public `priestley_taylor_alpha` (LAI0, param, mask0, nodata, cell_id0, nL0_in_L1, Upp_row_L1, Low_row_L1, Lef_col_L1, Rig_col_L1, priestley_taylor_alpha1)
Regionalization of priestley taylor alpha.
- subroutine, public `bulksurface_resistance` (LAI0, param, mask0, nodata, cell_id0, nL0_in_L1, Upp_row_L1, Low_row_L1, Lef_col_L1, Rig_col_L1, bulksurface_resistance1)
Regionalization of bulk surface resistance.

15.40.1 Detailed Description

TODO: add description.

This module sets up pet correction factor at level-1 based on LAI

Authors

Mehmet Cuneyd Demirel, Simon Stisen

Date

May 2017

15.40.2 Function/Subroutine Documentation

15.40.2.1 bulksurface_resistance()

```
subroutine, public mo_mpr_pet::bulksurface_resistance (
    real(dp), dimension(:, :, ), intent(in) LAI0,
    real(dp), intent(in) param,
    logical, dimension(:, :, ), intent(in) mask0,
    real(dp), intent(in) nodata,
    integer(i4), dimension(:, ), intent(in) cell_id0,
    integer(i4), dimension(:, ), intent(in) nL0_in_L1,
    integer(i4), dimension(:, ), intent(in) Upp_row_L1,
    integer(i4), dimension(:, ), intent(in) Low_row_L1,
    integer(i4), dimension(:, ), intent(in) Lef_col_L1,
    integer(i4), dimension(:, ), intent(in) Rig_col_L1,
    real(dp), dimension(:, :, ), intent(out) bulksurface_resistance1 )
```

Regionalization of bulk surface resistance.

estimation of bulk surface resistance Global parameters needed (see mhm_parameter.nml):

- param(1) = stomatal_resistance

Parameters

in	<i>real(dp), dimension(:, :) :: LAI0</i>	LAI at level-0
in	<i>real(dp) :: param</i>	- global parameter
in	<i>logical, dimension(:, :) :: mask0</i>	mask at level 0
in	<i>real(dp) :: nodata</i>	- nodata value
in	<i>integer(i4), dimension(:) :: cell_id0</i>	Cell ids of hi res field
in	<i>integer(i4), dimension(:) :: nL0_in_L1</i>	number of l0 cells within a l1 cell
in	<i>integer(i4), dimension(:) :: Upp_row_L1</i>	upper row of a l1 cell in l0 grid
in	<i>integer(i4), dimension(:) :: Low_row_L1</i>	lower row of a l1 cell in l0 grid
in	<i>integer(i4), dimension(:) :: Lef_col_L1</i>	left col of a l1 cell in l0 grid
in	<i>integer(i4), dimension(:) :: Rig_col_L1</i>	right col of a l1 cell in l0 grid
out	<i>real(dp), dimension(:, :) :: bulksurface_resistance1</i>	bulk surface resistance

Authors

Matthias Zink

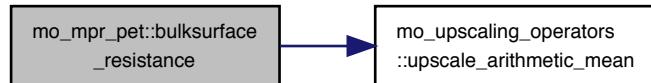
Date

Apr 2013

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

Referenced by `mo_multi_param_reg::mpr()`.

Here is the call graph for this function:



Here is the caller graph for this function:

15.40.2.2 `pet_correctbyasp()`

```

subroutine, public mo_mpr_pet::pet_correctbyasp (
    integer(i4), dimension(:), intent(in) Id0,
  
```

```

real(dp), dimension(:, ), intent(in) latitude_10,
real(dp), dimension(:, ), intent(in) Asp0,
real(dp), dimension(3), intent(in) param,
real(dp), intent(in) nodata,
real(dp), dimension(:, ), intent(out) fAsp0 )

```

correction of PET

Correction of PET based on L0 aspect data. Global parameters needed (see mhm_parameter.nml):

- param(1) = minCorrectionFactorPET
- param(2) = maxCorrectionFactorPET
- param(3) = aspectThresholdPET

Parameters

in	integer(i4), dimension(:) :: id0	Level 0 cell id
in	real(dp), dimension(:) :: latitude_10	latitude on L0
in	real(dp), dimension(:) :: Asp0	[degree] Aspect at Level 0
in	real(dp), dimension(3) :: param	process parameters
in	real(dp) :: nodata	- no data value
out	real(dp), dimension(:) :: fAsp0	PET correction for Aspect

Authors

Stephan Thober, Rohini Kumar

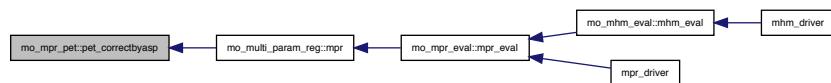
Date

Dec 2012

References mo_kind::i4.

Referenced by mo_multi_param_reg::mpr().

Here is the caller graph for this function:



15.40.2.3 pet_correctbylai()

```

subroutine, public mo_mpr_pet::pet_correctbylai (
    real(dp), dimension(5), intent(in) param,
    real(dp), intent(in) nodata,
    integer(i4), dimension(:, ), intent(in) LCOVER0,
    real(dp), dimension(:, :, ), intent(in) LAI0,
    logical, dimension(:, :, ), intent(in) mask0,
    integer(i4), dimension(:, ), intent(in) cell_id0,
    integer(i4), dimension(:, ), intent(in) upp_row_L1,

```

```

integer(i4), dimension(:), intent(in) low_row_L1,
integer(i4), dimension(:), intent(in) lef_col_L1,
integer(i4), dimension(:), intent(in) rig_col_L1,
integer(i4), dimension(:), intent(in) nL0_in_L1,
real(dp), dimension(:, :), intent(inout) L1_petLAIcorFactor )

```

estimate PET correction factor based on LAI at L1

estimate PET correction factor based on LAI at L1 for a given Leaf Area Index field. Global parameters needed (see mhm_parameter.nml): Process Case 5:

- param(1) = PET_a_forest
- param(2) = PET_a_impervious
- param(3) = PET_a_pervious
- param(4) = PET_b
- param(5) = PET_c Example DSF=PET_a+PET_b*(1-exp(PET_c*LAI)) Similar to the crop coefficient concept $Kc=a+b*(1-exp(c*LAI))$ by Allen, R. G., L. S. Pereira, D. Raes, and M. Smith (1998), Crop evapotranspiration - Guidelines for computing crop water requirements., FAO Irrigation and drainage paper 56. See Chapter 9, Equation 97 <http://www.fao.org/docrep/X0490E/x0490e0f.htm> Date: 17/5/2017

Parameters

in	real(dp), dimension(5) :: param	parameters
in	real(dp) :: nodata	- nodata value
in	integer(i4), dimension(:) :: LCOVER0	Land cover at level 0
in	real(dp), dimension(:, :) :: LAI0	LAI at level-0
in	logical, dimension(:, :) :: mask0	mask at L0
in	integer(i4), dimension(:) :: cell_id0	Cell ids of hi res field
in	integer(i4), dimension(:) :: upp_row_L1	Upper row of hi res block
in	integer(i4), dimension(:) :: low_row_L1	Lower row of hi res block
in	integer(i4), dimension(:) :: lef_col_L1	Left column of hi res block
in	integer(i4), dimension(:) :: rig_col_L1	Right column of hi res block
in	integer(i4), dimension(:) :: nL0_in_L1	Number of L0 cells within a L1 cel
in, out	real(dp), dimension(:, :) :: L1_petLAIcorFactor	pet cor factor at level-1

Authors

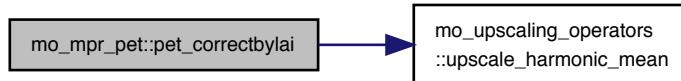
M. Cuneyd Demirel and Simon Stisen from GEUS.dk

Date

May. 2017

References `mo_upscaling_operators::upscale_harmonic_mean()`.Referenced by `mo_multi_param_reg::mpr()`.

Here is the call graph for this function:



Here is the caller graph for this function:

15.40.2.4 `priestley_taylor_alpha()`

```

subroutine, public mo_mpr_pet::priestley_taylor_alpha (
    real(dp), dimension(:, :, ), intent(in) LAI0,
    real(dp), dimension(:, ), intent(in) param,
    logical, dimension(:, :, ), intent(in) mask0,
    real(dp), intent(in) nodata,
    integer(i4), dimension(:, ), intent(in) cell_id0,
    integer(i4), dimension(:, ), intent(in) nL0_in_L1,
    integer(i4), dimension(:, ), intent(in) Upp_row_L1,
    integer(i4), dimension(:, ), intent(in) Low_row_L1,
    integer(i4), dimension(:, ), intent(in) Lef_col_L1,
    integer(i4), dimension(:, ), intent(in) Rig_col_L1,
    real(dp), dimension(:, :, ), intent(out) priestley_taylor_alpha1 )
  
```

Regionalization of priestley taylor alpha.

estimation of priestley taylor alpha Global parameters needed (see `mhm_parameter.nml`):

- `param(1) = PriestleyTaylorCoeff`
- `param(2) = PriestleyTaylorLAlcorr`

Parameters

in	<code>real(dp), dimension(:, :,) :: LAI0</code>	LAI at level-0
in	<code>real(dp), dimension(:,) :: param</code>	input parameter

Parameters

in	<i>logical, dimension(:, :) :: mask0</i>	mask at level 0
in	<i>real(dp) :: nodata</i>	- nodata value
in	<i>integer(i4), dimension(:) :: cell_id0</i>	Cell ids of hi res field
in	<i>integer(i4), dimension(:) :: nL0_in_L1</i>	number of l0 cells within a l1 cell
in	<i>integer(i4), dimension(:) :: Upp_row_L1</i>	upper row of a l1 cell in l0 grid
in	<i>integer(i4), dimension(:) :: Low_row_L1</i>	lower row of a l1 cell in l0 grid
in	<i>integer(i4), dimension(:) :: Lef_col_L1</i>	left col of a l1 cell in l0 grid
in	<i>integer(i4), dimension(:) :: Rig_col_L1</i>	right col of a l1 cell in l0 grid
out	<i>real(dp), dimension(:, :) :: priestley_taylor_alpha1</i>	bulk surface resistance

Authors

Matthias Zink

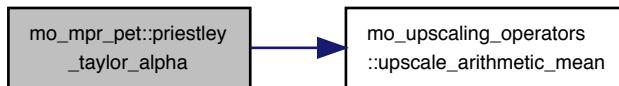
Date

Apr 2013

References `mo_upscaling_operators::upscale_arithmetic_mean()`.

Referenced by `mo_multi_param_reg::mpr()`.

Here is the call graph for this function:



Here is the caller graph for this function:



15.41 mo_mpr_read_config Module Reference

read mpr config

Functions/Subroutines

- subroutine, public `mp_r_ead_cnf` (`file_namelst, unamelst, file_namelst_param, unamelst_param`)
Read the general config of mpr.

15.41.1 Detailed Description

read mpr config

This module contains all mpr subroutines related to reading the mpr configuration from file.

Authors

Stephan Thober

Date

Aug 2015

15.41.2 Function/Subroutine Documentation

15.41.2.1 mpr_read_config()

```
subroutine, public mo_mpr_read_config::mpr_read_config (
    character(*), intent(in) file_namelist,
    integer, intent(in) unamelist,
    character(*), intent(in) file_namelist_param,
    integer, intent(in) unamelist_param )
```

Read the general config of mpr.

Depending on the variable mrm_coupling_config, the mRM config is either read from mrm.nml and parameters from mrm_parameter.nml or copied from mHM.

Parameters

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

Authors

Stephan Thober

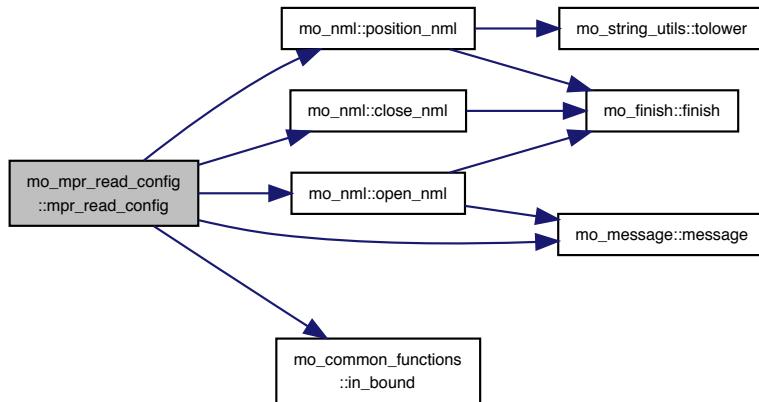
Date

Aug 2015

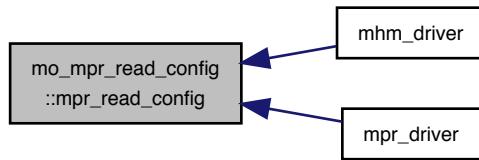
References mo_nml::close_nml(), mo_mpr_global_variables::dirgridded_lai, mo_common_constants::eps_dp, mo_mpr_global_variables::fracsealed_cityarea, mo_common_variables::global_parameters, mo_common_variables::global_parameters_name, mo_mpr_global_variables::horizondepth_mhm, mo_mpr_global_variables::iflag_soildb, mo_common_functions::in_bound(), mo_mpr_global_variables::inputformat_gridded_lai, mo_mpr_constants::maxgeounit, mo_common_constants::maxnobasins, mo_mpr_constants::maxnosoilhorizons, mo_message::message(), mo_common_variables::nbasins, mo_common_constants::ncolpars, mo_mpr_global_variables::ngeounits, mo_common_constants::nodata_dp, mo_mpr_global_variables::nsoilhorizons_mhm, mo_nml::open_nml(), mo_nml::position_nml(), mo_common_variables::processmatrix, mo_mpr_global_variables::tillagedepth, and mo_mpr_global_variables::timestep_lai_input.

Referenced by mhm_driver(), and mpr_driver().

Here is the call graph for this function:



Here is the caller graph for this function:



15.42 mo_mpr_restart Module Reference

reading and writing states, fluxes and configuration for restart of mHM.

Data Types

- interface [unpack_field_and_write](#)
TODO: add description.

Functions/Subroutines

- subroutine, public [write_mpr_restart_files](#) (OutPath)
write restart files for each basin
- subroutine, public [write_eff_params](#) (mask1, s1, e1, rows1, cols1, soil1, lscenes, lais, nc)
TODO: add description.
- subroutine [unpack_field_and_write_1d_i4](#) (nc, var_name, var_dims, fill_value, data, mask, var_long_name)

- subroutine `unpack_field_and_write_1d_dp` (nc, var_name, var_dims, fill_value, data, mask, var_long_name)
- subroutine `unpack_field_and_write_2d_dp` (nc, var_name, var_dims, fill_value, data, mask, var_long_name)
- subroutine `unpack_field_and_write_3d_dp` (nc, var_name, var_dims, fill_value, data, mask, var_long_name)

15.42.1 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

15.42.2 Function/Subroutine Documentation

15.42.2.1 `unpack_field_and_write_1d_dp()`

```
subroutine mo_mpr_restart::unpack_field_and_write_1d_dp (
    type(ncdataset), intent(inout) nc,
    character(*), intent(in) var_name,
    type(ncdimension), dimension(:), intent(in) var_dims,
    real(dp), intent(in) fill_value,
    real(dp), dimension(:), intent(in) data,
    logical, dimension(:, :), intent(in) mask,
    character(*), intent(in), optional var_long_name )
```

References `mo_kind::dp`.

15.42.2.2 `unpack_field_and_write_1d_i4()`

```
subroutine mo_mpr_restart::unpack_field_and_write_1d_i4 (
    type(ncdataset), intent(inout) nc,
    character(*), intent(in) var_name,
    type(ncdimension), dimension(:), intent(in) var_dims,
    integer(i4), intent(in) fill_value,
    integer(i4), dimension(:), intent(in) data,
    logical, dimension(:, :), intent(in) mask,
    character(*), intent(in), optional var_long_name )
```

References `mo_kind::i4`.

15.42.2.3 unpack_field_and_write_2d_dp()

```
subroutine mo_mpr_restart::unpack_field_and_write_2d_dp (
    type(ncdataset), intent(inout) nc,
    character(*), intent(in) var_name,
    type(ncdimension), dimension(:), intent(in) var_dims,
    real(dp), intent(in) fill_value,
    real(dp), dimension(:, :), intent(in) data,
    logical, dimension(:, :), intent(in) mask,
    character(*), intent(in), optional var_long_name )
```

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

15.42.2.4 unpack_field_and_write_3d_dp()

```
subroutine mo_mpr_restart::unpack_field_and_write_3d_dp (
    type(ncdataset), intent(inout) nc,
    character(*), intent(in) var_name,
    type(ncdimension), dimension(:), intent(in) var_dims,
    real(dp), intent(in) fill_value,
    real(dp), dimension(:, :, :), intent(in) data,
    logical, dimension(:, :, :), intent(in) mask,
    character(*), intent(in), optional var_long_name )
```

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

15.42.2.5 write_eff_params()

```
subroutine, public mo_mpr_restart::write_eff_params (
    logical, dimension(:, :, :), intent(in), allocatable mask1,
    integer(i4), intent(in) s1,
    integer(i4), intent(in) e1,
    type(ncdimension), intent(in) rows1,
    type(ncdimension), intent(in) cols1,
    type(ncdimension), intent(in) soil1,
    type(ncdimension), intent(in) lcscenes,
    type(ncdimension), intent(in) lais,
    type(ncdataset), intent(inout) nc )
```

TODO: add description.

TODO: add description

Parameters

in	<i>logical, dimension(:, :, :)</i> :: <i>mask1</i>	mask at level 1
in	<i>integer(i4)</i> :: <i>s1</i>	start index at level 1
in	<i>integer(i4)</i> :: <i>e1</i>	end index at level 1
in	<i>type(NcDimension)</i> :: <i>rows1, cols1, soil1, lcscenes, lais</i>	
in	<i>type(NcDimension)</i> :: <i>rows1, cols1, soil1, lcscenes, lais</i>	
in	<i>type(NcDimension)</i> :: <i>rows1, cols1, soil1, lcscenes, lais</i>	
in	<i>type(NcDimension)</i> :: <i>rows1, cols1, soil1, lcscenes, lais</i>	
in	<i>type(NcDimension)</i> :: <i>rows1, cols1, soil1, lcscenes, lais</i>	
in, out	<i>type(NcDataset)</i> :: <i>nc</i>	

Authors

Robert Scheweppe

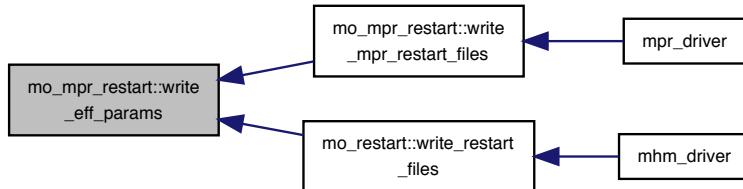
Date

Jun 2018

References `mo_kind::i4`, `mo_mpr_global_variables::l1_aeroresist`, `mo_mpr_global_variables::l1_alpha`, `mo_mpr_global_variables::l1_degday`, `mo_mpr_global_variables::l1_degdayinc`, `mo_mpr_global_variables::l1_degdaymax`, `mo_mpr_global_variables::l1_degdaynopre`, `mo_mpr_global_variables::l1_fasp`, `mo_mpr_global_variables::l1_froots`, `mo_mpr_global_variables::l1_fsealed`, `mo_mpr_global_variables::l1_harsamcoeff`, `mo_mpr_global_variables::l1_kbaseflow`, `mo_mpr_global_variables::l1_kfastflow`, `mo_mpr_global_variables::l1_kperco`, `mo_mpr_global_variables::l1_kslowflow`, `mo_mpr_global_variables::l1_maxinter`, `mo_mpr_global_variables::l1_petlaicfactor`, `mo_mpr_global_variables::l1_prietaryalpha`, `mo_mpr_global_variables::l1_sealedthresh`, `mo_mpr_global_variables::l1_soilmoistexp`, `mo_mpr_global_variables::l1_soilmoistfc`, `mo_mpr_global_variables::l1_soilmoistsat`, `mo_mpr_global_variables::l1_surfresist`, `mo_mpr_global_variables::l1_tempthresh`, `mo_mpr_global_variables::l1_unsatthresh`, `mo_mpr_global_variables::l1_wiltingpoint`, `mo_common_variables::lc_year_end`, `mo_common_variables::lc_year_start`, `mo_common_constants::nodata_dp`, `mo_common_constants::nodata_i4`, and `mo_common_variables::processmatrix`.

Referenced by `write_mpr_restart_files()`, and `mo_restart::write_restart_files()`.

Here is the caller graph for this function:



15.42.2.6 `write_mpr_restart_files()`

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

write restart files for each basin

write restart files for each basin. For each basin three restart files are written. These are `xxx_states.nc`, `xxx_L11_config.nc`, and `xxx_config.nc` (xxx being the three digit basin index). If a variable is added here, it should also be added in the read restart routines below. ADDITIONAL INFORMATION `write_restart`

Parameters

in	<code>character(256), dimension(:) :: OutPath</code>	Output Path for each basin
----	--	----------------------------

Authors

Stephan Thober

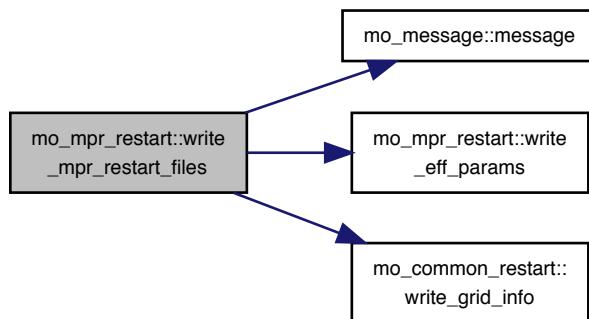
Date

Jun 2014

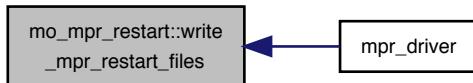
References mo_kind::i4, mo_common_variables::level1, mo_message::message(), mo_mpr_global_variables::nlai, mo_common_variables::nlcoverscene, mo_mpr_global_variables::nsoilhorizons_mhm, write_eff_params(), and mo_common_restart::write_grid_info().

Referenced by mpr_driver().

Here is the call graph for this function:



Here is the caller graph for this function:



15.43 mo_mpr_runoff Module Reference

multiscale parameter regionalization for runoff generation

Functions/Subroutines

- subroutine, public **mp_runoff** (LCOVER0, mask0, SMs_FC0, slope_emp0, KsVar_H0, param, cell_id0, upp_low_row_L1, low_row_L1, lef_col_L1, rig_col_L1, nL0_in_L1, c2TSTu, L1_HL1, L1_K0, L1_K1, L1_alpha)
multiscale parameter regionalization for runoff parameters

15.43.1 Detailed Description

multiscale parameter regionalization for runoff generation

This contains the routine for multiscale parameter regionalization of the runoff parametrization.

Authors

Stephan Thober, Rohini Kumar

Date

Dec 2012

15.43.2 Function/Subroutine Documentation

15.43.2.1 mpr_runoff()

```
subroutine, public mo_mpr_runoff::mpr_runoff (
    integer(i4), dimension(:), intent(in) LCOVER0,
    logical, dimension(:, :), intent(in) mask0,
    real(dp), dimension(:), intent(in) SMs_FCO,
    real(dp), dimension(:), intent(in) slope_emp0,
    real(dp), dimension(:), intent(in) KsVar_H0,
    real(dp), dimension(5), intent(in) param,
    integer(i4), dimension(:), intent(in) cell_id0,
    integer(i4), dimension(:), intent(in) upp_row_L1,
    integer(i4), dimension(:), intent(in) low_row_L1,
    integer(i4), dimension(:), intent(in) lef_col_L1,
    integer(i4), dimension(:), intent(in) rig_col_L1,
    integer(i4), dimension(:), intent(in) nL0_in_L1,
    real(dp), intent(in) c2TSTu,
    real(dp), dimension(:), intent(out) L1_HL1,
    real(dp), dimension(:), intent(out) L1_K0,
    real(dp), dimension(:), intent(out) L1_K1,
    real(dp), dimension(:), intent(out) L1_alpha )
```

multiscale parameter regionalization for runoff parameters

Perform the multiscale parameter regionalization for runoff global parameters (see mhm_parameter.nml). These are the following five parameters:

- param(1) = interflowStorageCapacityFactor
- param(2) = interflowRecession_slope
- param(3) = fastInterflowRecession_forest
- param(4) = slowInterflowRecession_Ks
- param(5) = exponentSlowInterflow

Parameters

in	<i>integer(i4), dimension(:) :: LCOVER0</i>	land cover at level 0
in	<i>logical, dimension(:, :) :: mask0</i>	mask at Level 0
in	<i>real(dp), dimension(:) :: SMs_FCO</i>	[-] soil moisture deficit from field

Parameters

in	<i>real(dp), dimension(:) :: slope_emp0</i>	empirical quantile values F(slope)
in	<i>real(dp), dimension(:) :: KsVar_H0</i>	[‐] relative variability of saturated
in	<i>real(dp), dimension(5) :: param</i>	global parameters
in	<i>integer(i4), dimension(:) :: cell_id0</i>	Cell ids of hi res field
in	<i>integer(i4), dimension(:) :: upp_row_L1</i>	Upper row of hi res block
in	<i>integer(i4), dimension(:) :: low_row_L1</i>	Lower row of hi res block
in	<i>integer(i4), dimension(:) :: lef_col_L1</i>	Left column of hi res block
in	<i>integer(i4), dimension(:) :: rig_col_L1</i>	Right column of hi res block
in	<i>integer(i4), dimension(:) :: nL0_in_L1</i>	Number of L0 cells within a L1 cell
in	<i>real(dp) :: c2TSTu</i>	unit transformations
out	<i>real(dp), dimension(:) :: L1_HL1</i>	[10 ^{‐3} m] Threshhold water depth
out	<i>real(dp), dimension(:) :: L1_K0</i>	[10 ^{‐3} m] Recession coefficient
out	<i>real(dp), dimension(:) :: L1_K1</i>	[10 ^{‐3} m] Recession coefficient
out	<i>real(dp), dimension(:) :: L1_alpha</i>	[1] Exponent for the upper reservoir

Authors

Stephan Thober, Rohini Kumar

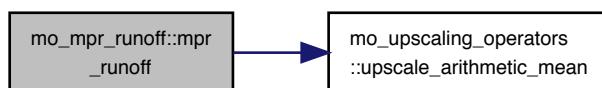
Date

Dec 2012

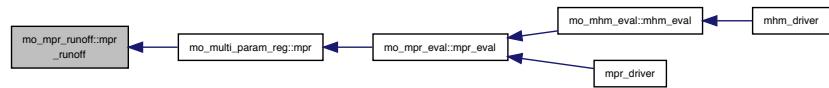
References mo_common_constants::nodata_dp, mo_common_constants::nodata_i4, and mo_upscaling::operators::upscale_arithmetic_mean().

Referenced by mo_multi_param_reg::mpr().

Here is the call graph for this function:



Here is the caller graph for this function:



15.44 mo_mpr_smhorizons Module Reference

setting up the soil moisture horizons

Functions/Subroutines

- subroutine, public `mpr_smhorizons` (param, processMatrix, iFlag_soil, nHorizons_mHM, HorizonDepth, LCOVER0, soilID0, nHorizons, nTillHorizons, thetaS_till, thetaFC_till, thetaPW_till, thetaS, thetaFC, thetaPW, Wd, Db, DbM, RZdepth, mask0, cell_id0, upp_row_L1, low_row_L1, lef_col_L1, rig_col_L1, nL0_in_L1, L1_beta, L1_SMs, L1_FC, L1_PW, L1_fRoots)
upscale soil moisture horizons

15.44.1 Detailed Description

setting up the soil moisture horizons

This module sets up the soil moisture horizons

Authors

Stephan Thober, Rohini Kumar

Date

Dec 2012

15.44.2 Function/Subroutine Documentation

15.44.2.1 mpr_smhorizons()

```
subroutine, public mo_mpr_smhorizons::mpr_smhorizons (
    real(dp), dimension(:, :, :), intent(in) param,
    integer(i4), dimension(:, :, :), intent(in) processMatrix,
    integer(i4), intent(in) iFlag_soil,
    integer(i4), intent(in) nHorizons_mHM,
    real(dp), dimension(:, :, :), intent(in) HorizonDepth,
    integer(i4), dimension(:, :, :), intent(in) LCOVER0,
    integer(i4), dimension(:, :, :), intent(in) soilID0,
    integer(i4), dimension(:, :, :), intent(in) nHorizons,
    integer(i4), dimension(:, :, :), intent(in) nTillHorizons,
    real(dp), dimension(:, :, :, :), intent(in) thetaS_till,
    real(dp), dimension(:, :, :, :), intent(in) thetaFC_till,
    real(dp), dimension(:, :, :, :), intent(in) thetaPW_till,
    real(dp), dimension(:, :, :, :), intent(in) thetaS,
    real(dp), dimension(:, :, :, :), intent(in) thetaFC,
    real(dp), dimension(:, :, :, :), intent(in) thetaPW,
    real(dp), dimension(:, :, :, :), intent(in) Wd,
    real(dp), dimension(:, :, :, :), intent(in) Db,
    real(dp), dimension(:, :, :, :), intent(in) DbM,
    real(dp), dimension(:, :, :, :), intent(in) RZdepth,
    logical, dimension(:, :, :, :), intent(in) mask0,
    integer(i4), dimension(:, :, :, :), intent(in) cell_id0,
    integer(i4), dimension(:, :, :, :), intent(in) upp_row_L1,
```

```

integer(i4), dimension(:), intent(in) low_row_L1,
integer(i4), dimension(:), intent(in) lef_col_L1,
integer(i4), dimension(:), intent(in) rig_col_L1,
integer(i4), dimension(:), intent(in) nL0_in_L1,
real(dp), dimension(:, :), intent(inout) L1_beta,
real(dp), dimension(:, :), intent(inout) L1_SMS,
real(dp), dimension(:, :), intent(inout) L1_FC,
real(dp), dimension(:, :), intent(inout) L1_PW,
real(dp), dimension(:, :), intent(inout) L1_fRoots )

```

upscale soil moisture horizons

calculate soil properties at the level 1. Global parameters needed (see mhmm_parameter.nml):

- param(1) = rootFractionCoefficient_forest
- param(2) = rootFractionCoefficient_impermeous
- param(3) = rootFractionCoefficient_pervious
- param(4) = infiltrationShapeFactor

Parameters

in	real(dp), dimension(:) :: param	parameters
in	integer(i4), dimension(:, :) :: processMatrix	- matrix specifying user defined processes
in	integer(i4) :: iFlag_soil	- flags for handling multiple soil databases
in	integer(i4) :: nHorizons_mHM	- number of horizons to model
in	real(dp), dimension(:) :: HorizonDepth	[10^-3 m] horizon depth from surface, positive downwards
in	integer(i4), dimension(:) :: LCOVER0	Land cover at level 0
in	integer(i4), dimension(:, :) :: soilID0	soil ID at level 0
in	integer(i4), dimension(:) :: nHorizons	horizons per soil type
in	integer(i4), dimension(:) :: nTillHorizons	Number of Tillage horizons
in	real(dp), dimension(:, :, :) :: thetaS_till	saturated water content of soil horizons upto tillage depth, f(OM, management)
in	real(dp), dimension(:, :, :) :: thetaFC_till	Field capacity of tillage layers; LUC dependent, f(OM, management)
in	real(dp), dimension(:, :, :) :: thetaPW_till	Permanent wilting point of tillage layers; LUC dependent, f(OM, management)
in	real(dp), dimension(:, :) :: thetaS	saturated water content of soil horizons after tillage depth
in	real(dp), dimension(:, :) :: thetaFC	Field capacity of deeper layers
in	real(dp), dimension(:, :) :: thetaPW	Permanent wilting point of deeper layers
in	real(dp), dimension(:, :, :) :: Wd	weights of mHM Horizons according to horizons provided in soil database
in	real(dp), dimension(:, :, :) :: Db	Bulk density
in	real(dp), dimension(:, :) :: DbM	mineral Bulk density
in	real(dp), dimension(:) :: RZdepth	[mm] Total soil depth
in	logical, dimension(:, :) :: mask0	mask at L0
in	integer(i4), dimension(:) :: cell_id0	Cell ids of hi res field
in	integer(i4), dimension(:) :: upp_row_L1	Upper row of hi res block
in	integer(i4), dimension(:) :: low_row_L1	Lower row of hi res block
in	integer(i4), dimension(:) :: lef_col_L1	Left column of hi res block
in	integer(i4), dimension(:) :: rig_col_L1	Right column of hi res block

Parameters

in	<i>integer(i4), dimension(:) :: nL0_in_L1</i>	Number of L0 cells within a L1 cel
in, out	<i>real(dp), dimension(:, :) :: L1_beta</i>	Parameter that determines the relative contribution to SM, upscaled Bulk density
in, out	<i>real(dp), dimension(:, :) :: L1_SMS</i>	[10^-3 m] depth of saturated SM cont
in, out	<i>real(dp), dimension(:, :) :: L1_FC</i>	[10^-3 m] field capacity
in, out	<i>real(dp), dimension(:, :) :: L1_PW</i>	[10^-3 m] permanent wilting point
in, out	<i>real(dp), dimension(:, :) :: L1_fRoots</i>	fraction of roots in soil horizons

Authors

Luis Samaniego, Rohini Kumar, Stephan Thober

Date

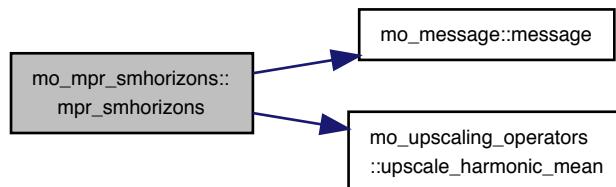
Dec 2012

in this case the second dimension of soilld0 = 1

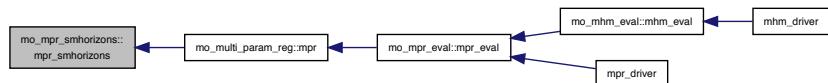
References `mo_message::message()`, `mo_common_constants::nodata_dp`, and `mo_upscaling_operators::upscale_harmonic_mean()`.

Referenced by `mo_multi_param_reg::mpr()`.

Here is the call graph for this function:



Here is the caller graph for this function:



15.45 mo_mpr_soilmoist Module Reference

Multiscale parameter regionalization (MPR) for soil moisture.

Functions/Subroutines

- subroutine, public [mpr_sm](#) (param, is_present, nHorizons, nTillHorizons, sand, clay, DbM, ID0, soilld0, L← Cover0, thetaS_till, thetaFC_till, thetaPW_till, thetaS, thetaFC, thetaPW, Ks, Db, KsVar_H0, KsVar_V0, S← Ms_FC0)

multiscale parameter regionalization for soil moisture
- elemental pure subroutine [pwp](#) (Genu_Mual_n, Genu_Mual_alpha, thetaS, thetaPWP)

Permanent Wilting point.
- elemental pure subroutine [field_cap](#) (thetaFC, Ks, thetaS, Genu_Mual_n)

calculates the field capacity
- subroutine [genuchten](#) (thetaS, Genu_Mual_n, Genu_Mual_alpha, param, sand, clay, Db)

calculates the Genuchten shape parameter
- subroutine [hydro_cond](#) (KS, param, sand, clay)

calculates the hydraulic conductivity Ks

15.45.1 Detailed Description

Multiscale parameter regionalization (MPR) for soil moisture.

This module contains all routines required for parametrizing soil moisture processes.

Authors

Stephan Thober, Rohini Kumar

Date

Dec 2012

15.45.2 Function/Subroutine Documentation

15.45.2.1 [field_cap\(\)](#)

```
elemental pure subroutine mo_mpr_soilmoist::field_cap (
    real(dp), intent(out) thetaFC,
    real(dp), intent(in) Ks,
    real(dp), intent(in) thetaS,
    real(dp), intent(in) Genu_Mual_n )
```

calculates the field capacity

estimate Field capacity; FC – Flux based approach (Twarakavi, et. al. 2009, WRR) According to the above reference FC is defined as the soil water content at which the drainage from a profile ceases under natural conditions. Since drainage from a soil profile in a simulation never becomes zero, we assume that drainage ceases when the bottom flux from the soil reaches a value that is equivalent to the minimum amount of precipitation that could be recorded (i.e. 0.01 cm/d == 1 mm/d). It is assumed that ThetaR = 0.0_dp ADDITIONAL INFORMATION Twarakavi, et. al. 2009, WRR

Parameters

out	<code>real(dp) :: thetaFC</code>	- Field capacity
in	<code>real(dp) :: Ks</code>	- saturated hydraulic conductivity
in	<code>real(dp) :: thetaS</code>	- saturated water content
in	<code>real(dp) :: Genu_Mual_n</code>	- Genuchten shape parameter

Authors

Stephan Thober, Rohini Kumar

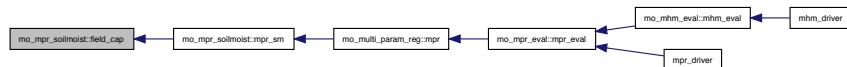
Date

Dec 2012

References mo_mpr_constants::field_cap_c1, and mo_mpr_constants::field_cap_c2.

Referenced by mpr_sm().

Here is the caller graph for this function:



15.45.2.2 genuchten()

```

subroutine mo_mpr_soilmoist::genuchten (
    real(dp), intent(out) thetaS,
    real(dp), intent(out) Genu_Mual_n,
    real(dp), intent(out) Genu_Mual_alpha,
    real(dp), dimension(6), intent(in) param,
    real(dp), intent(in) sand,
    real(dp), intent(in) clay,
    real(dp), intent(in) Db )
  
```

calculates the Genuchten shape parameter

estimate SMs_till & van Genuchten's shape parameter (n) (Zacharias et al, 2007, soil Phy.) Global parameters needed (see mhm_parameter.nml):

- param(1) = PTF_lower66_5_constant
- param(2) = PTF_lower66_5_clay
- param(3) = PTF_lower66_5_Db
- param(4) = PTF_higher66_5_constant
- param(5) = PTF_higher66_5_clay
- param(6) = PTF_higher66_5_Db ADDITIONAL INFORMATION Zacharias et al, 2007, soil Phy.

Parameters

out	real(dp) :: thetaS	- saturated water content
out	real(dp) :: Genu_Mual_n	- van Genuchten shape parameter
out	real(dp) :: Genu_Mual_alpha	- van Genuchten shape parameter
in	real(dp), dimension(6) :: param	parameters
in	real(dp) :: sand	- [%] sand content
in	real(dp) :: clay	- [%] clay content
in	real(dp) :: Db	- [10^3 kg/m3] bulk density

Authors

Stephan Thober, Rohini Kumar

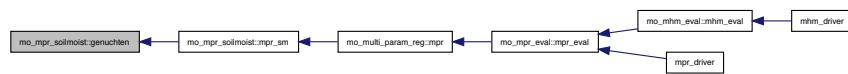
Date

Dec 2012

References mo_mpr_constants::vgenuchten_sandtresh, mo_mpr_constants::vgenuchtenn_c1, mo_mpr_constants::vgenuchtenn_c10, mo_mpr_constants::vgenuchtenn_c11, mo_mpr_constants::vgenuchtenn_c12, mo_mpr_constants::vgenuchtenn_c13, mo_mpr_constants::vgenuchtenn_c14, mo_mpr_constants::vgenuchtenn_c15, mo_mpr_constants::vgenuchtenn_c16, mo_mpr_constants::vgenuchtenn_c17, mo_mpr_constants::vgenuchtenn_c18, mo_mpr_constants::vgenuchtenn_c2, mo_mpr_constants::vgenuchtenn_c3, mo_mpr_constants::vgenuchtenn_c4, mo_mpr_constants::vgenuchtenn_c5, mo_mpr_constants::vgenuchtenn_c6, mo_mpr_constants::vgenuchtenn_c7, mo_mpr_constants::vgenuchtenn_c8, and mo_mpr_constants::vgenuchtenn_c9.

Referenced by mpr_sm().

Here is the caller graph for this function:



15.45.2.3 hydro_cond()

```

subroutine mo_mpr_soilmoist::hydro_cond (
    real(dp), intent(out) KS,
    real(dp), dimension(4), intent(in) param,
    real(dp), intent(in) sand,
    real(dp), intent(in) clay )
  
```

calculates the hydraulic conductivity Ks

By default save this value of Ks, particularly for the deeper layers where OM content plays relatively low or no role
Global parameters needed (see mhm_parameter.nml):

- param(1) = PTF_Ks_constant
 - param(2) = PTF_Ks_sand
 - param(3) = PTF_Ks_clay
 - param(4) = PTF_Ks_curveSlope
- ADDITIONAL INFORMATION Written, Stephan Thober, Dec 2012

Parameters

out	<i>real(dp)</i> :: KS	
in	<i>real(dp)</i> , dimension(4) :: param	
in	<i>real(dp)</i> :: sand	- [%] sand content
in	<i>real(dp)</i> :: clay	- [%] clay content

Authors

Stephan Thober, Rohini Kumar

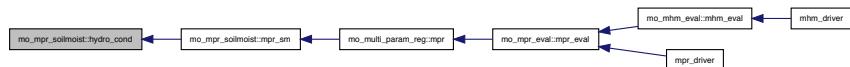
Date

Dec 2012

References `mo_mpr_constants::ks_c`.

Referenced by `mpr_sm()`.

Here is the caller graph for this function:



15.45.2.4 mpr_sm()

```

subroutine, public mo_mpr_soilmoist::mpr_sm (
    real(dp), dimension(13), intent(in) param,
    integer(i4), dimension(:, ), intent(in) is_present,
    integer(i4), dimension(:, ), intent(in) nHorizons,
    integer(i4), dimension(:, ), intent(in) nTillHorizons,
    real(dp), dimension(:, :, ), intent(in) sand,
    real(dp), dimension(:, :, ), intent(in) clay,
    real(dp), dimension(:, :, ), intent(in) DbM,
    integer(i4), dimension(:, ), intent(in) ID0,
    integer(i4), dimension(:, :, ), intent(in) soilId0,
    integer(i4), dimension(:, ), intent(in) LCover0,
    real(dp), dimension(:, :, :, ), intent(out) thetaS_till,
    real(dp), dimension(:, :, :, ), intent(out) thetaFC_till,
    real(dp), dimension(:, :, :, ), intent(out) thetaPW_till,
    real(dp), dimension(:, :, :, ), intent(out) thetaS,
    real(dp), dimension(:, :, :, ), intent(out) thetaFC,
    real(dp), dimension(:, :, :, ), intent(out) thetaPW,
    real(dp), dimension(:, :, :, ), intent(out) Ks,
    real(dp), dimension(:, :, :, ), intent(out) Db,
    real(dp), dimension(:, ), intent(out) KsVar_H0,
    real(dp), dimension(:, ), intent(out) KsVar_V0,
    real(dp), dimension(:, ), intent(out) SMs_FCO )

```

multiscale parameter regionalization for soil moisture

This subroutine is a wrapper around all soil moisture parameter routines. This subroutine requires 13 parameters. These parameters have to correspond to the parameters in the original parameter array at the following locations: 10-12, 13-18, 27-30. Global parameters needed (see `mhm_parameter.nml`):

- param(1) = orgMatterContent_forest
- param(2) = orgMatterContent_impervious
- param(3) = orgMatterContent_pervious
- param(4) = PTF_lower66_5_constant

- param(5) = PTF_lower66_5_clay
- param(6) = PTF_lower66_5_Db
- param(7) = PTF_higher66_5_constant
- param(8) = PTF_higher66_5_clay
- param(9) = PTF_higher66_5_Db
- param(10) = PTF_Ks_constant
- param(11) = PTF_Ks_sand
- param(12) = PTF_Ks_clay
- param(13) = PTF_Ks_curveSlope

Parameters

in	<i>real(dp), dimension(13) :: param</i>	global parameters
in	<i>integer(i4), dimension(:) :: is_present</i>	indicates whether soiltype is present
in	<i>integer(i4), dimension(:) :: nHorizons</i>	Number of Horizons per soiltype
in	<i>integer(i4), dimension(:) :: nTillHorizons</i>	Number of Tillage Horizons
in	<i>real(dp), dimension(:, :) :: sand</i>	sand content
in	<i>real(dp), dimension(:, :) :: clay</i>	clay content
in	<i>real(dp), dimension(:, :) :: DbM</i>	mineral Bulk density
in	<i>integer(i4), dimension(:) :: ID0</i>	cell ids at level 0
in	<i>integer(i4), dimension(:, :) :: soilld0</i>	soil ids at level 0
in	<i>integer(i4), dimension(:) :: LCOVER0</i>	land cover ids at level 0
out	<i>real(dp), dimension(:, :, :) :: thetaS_till</i>	saturated soil moisture tillage layer
out	<i>real(dp), dimension(:, :, :) :: thetaFC_till</i>	field capacity tillage layer
out	<i>real(dp), dimension(:, :, :) :: thetaPW_till</i>	permanent wilting point tillage layer
out	<i>real(dp), dimension(:, :) :: thetaS</i>	saturated soil moisture
out	<i>real(dp), dimension(:, :) :: thetaFC</i>	field capacity
out	<i>real(dp), dimension(:, :) :: thetaPW</i>	permanent wilting point
out	<i>real(dp), dimension(:, :, :) :: Ks</i>	saturated hydraulic conductivity
out	<i>real(dp), dimension(:, :, :) :: Db</i>	Bulk density
out	<i>real(dp), dimension(:) :: KsVar_H0</i>	rel. var. of Ks for horizontal flow
out	<i>real(dp), dimension(:) :: KsVar_V0</i>	rel. var. of Ks for vertical flow
out	<i>real(dp), dimension(:) :: SMs_FCO</i>	soil moisture deficit from field cap. w.r.t to saturation

Authors

Stephan Thober, Rohini Kumar

Date

Dec 2012

here = ncells0

in this case the second dimension of soilld0 = 1

non-till

till layers

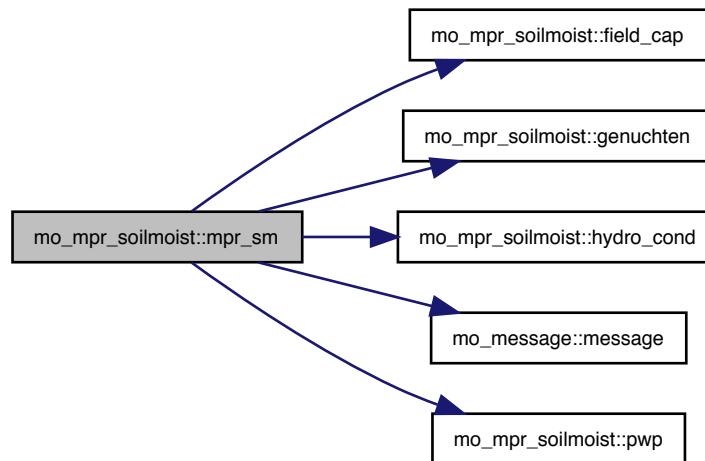
HORIZON

SOIL TYPE

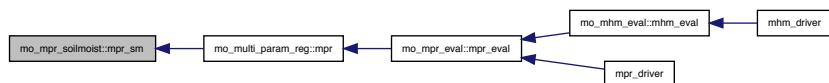
References mo_mpr_constants::bulkdens_orgmatter, field_cap(), genuchten(), hydro_cond(), mo_mpr_global::variables::iflag_soildb, mo_message::message(), mo_common_constants::nodata_dp, mo_common_constants::nodata_i4, and pwp().

Referenced by mo_multi_param_reg::mpr().

Here is the call graph for this function:



Here is the caller graph for this function:



15.45.2.5 pwp()

```

elemental pure subroutine mo_mpr_soilmoist::pwp (
    real(dp), intent(in) Genu_Mual_n,
    real(dp), intent(in) Genu_Mual_alpha,
    real(dp), intent(in) thetaS,
    real(dp), intent(out) thetaPWP )

```

Permanent Wilting point.

This subroutine calculates the permanent wilting point according to Zacharias et al. (2007, Soil Phy.) and using van Genuchten 1980's equation. For the water retention curve at a matrix potential of -1500 kPa, it is assumed that thetaR = 0. ADDITIONAL INFORMATION Zacharias et al. 2007, Soil Phy.

Parameters

in	<i>real(dp) :: Genu_Mual_n</i>	- Genuchten shape parameter
in	<i>real(dp) :: Genu_Mual_alpha</i>	- Genuchten shape parameter
in	<i>real(dp) :: thetaS</i>	- saturated water content
out	<i>real(dp) :: thetaPWP</i>	- Permanent Wilting point

Authors

Stephan Thober, Rohini Kumar

Date

Dec, 2012

References mo_mpr_constants::pwp_c, and mo_mpr_constants::pwp_matpot_theta.

Referenced by mpr_sm().

Here is the caller graph for this function:



15.46 mo_mpr_startup Module Reference

Startup procedures for mHM.

Functions/Subroutines

- subroutine, public [mpr_initialize](#)
Initialize main mHM variables.
- subroutine [IO_check_input](#) (iBasin)
Check for errors in L0 input data.
- subroutine [IO_variable_init](#) (iBasin)
level 0 variable initialization
- subroutine, public [init_eff_params](#) (ncells1)
Allocation of space for mHM related L1 and L11 variables.

15.46.1 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

15.46.2 Function/Subroutine Documentation

15.46.2.1 init_eff_params()

```
subroutine, public mo_mpr_startup::init_eff_params (
    integer(i4), intent(in) ncells1 )
```

Allocation of space for mHM related L1 and L11 variables.

Allocation of space for mHM related L1 and L11 variables (e.g., states, fluxes, and parameters) for a given basin. Variables allocated here is defined in them [mo_global_variables.f90](#) file. After allocating any variable in this routine, initialize them in the following `variables_default_init` subroutine:

Parameters

in	integer(i4) :: ncells1	
----	------------------------	--

Authors

Rohini Kumar

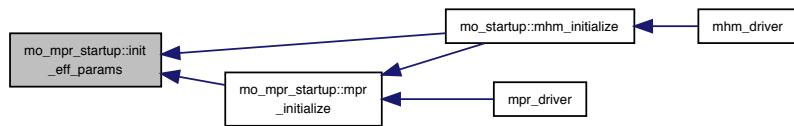
Date

Jan 2013

References `mo_mpr_global_variables::l1_aeroresist`, `mo_mpr_global_variables::l1_alpha`, `mo_mpr_global_variables::l1_deggday`, `mo_mpr_global_variables::l1_deggdayinc`, `mo_mpr_global_variables::l1_deggdaymax`, `mo_mpr_global_variables::l1_deggdaynopre`, `mo_mpr_global_variables::l1_fasp`, `mo_mpr_global_variables::l1_froots`, `mo_mpr_global_variables::l1_fsealed`, `mo_mpr_global_variables::l1_harsamcoeff`, `mo_mpr_global_variables::l1_jarvis_thresh_c1`, `mo_mpr_global_variables::l1_karstloss`, `mo_mpr_global_variables::l1_kbaseflow`, `mo_mpr_global_variables::l1_kfastflow`, `mo_mpr_global_variables::l1_kperco`, `mo_mpr_global_variables::l1_kslowflow`, `mo_mpr_global_variables::l1_maxinter`, `mo_mpr_global_variables::l1_petlaicorfactor`, `mo_mpr_global_variables::l1_prietayalpha`, `mo_mpr_global_variables::l1_sealedthresh`, `mo_mpr_global_variables::l1_soilmoistexp`, `mo_mpr_global_variables::l1_soilmoistfc`, `mo_mpr_global_variables::l1_soilmoistsat`, `mo_mpr_global_variables::l1_surfr Resist`, `mo_mpr_global_variables::l1_tempthresh`, `mo_mpr_global_variables::l1_unsatthresh`, `mo_mpr_global_variables::l1_wiltingpoint`, `mo_mpr_global_variables::nlai`, `mo_common_variables::nlcoverscene`, `mo_mpr_global_variables::nsoilhorizons_mhm`, `mo_common_constants::p1_initstatefluxes`, and `mo_common_constants::yearmonths_i4`.

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

Here is the caller graph for this function:



15.46.2.2 IO_check_input()

```
subroutine mo_mpr_startup::IO_check_input (
    integer(i4), intent(in) iBasin )
```

Check for errors in L0 input data.

Check for possible errors in input data (morphological and land cover) at level-0

Parameters

in	integer(i4) :: iBasin	basin id
----	-----------------------	----------

Authors

Rohini Kumar

Date

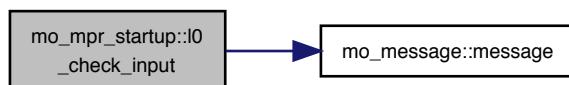
Jan 2013

by default; when iFlag_soilDB = 0

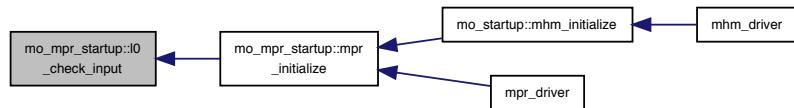
References mo_common_constants::eps_dp, mo_mpr_global_variables::iflag_soildb, mo_mpr_global_variables:::l0_asp, mo_common_variables::l0_elev, mo_mpr_global_variables::l0_geounit, mo_mpr_global_variables::l0_gridded_lai, mo_common_variables::l0_lcover, mo_mpr_global_variables::l0_slope, mo_mpr_global_variables::l0_soilid, mo_common_variables::level0, mo_message::message(), mo_message::message_text, mo_common_variables::nlcoverscene, mo_mpr_global_variables::nsoilhorizons_mhm, and mo_mpr_global_variables::timestep_lai_input.

Referenced by mpr_initialize().

Here is the call graph for this function:



Here is the caller graph for this function:



15.46.2.3 I0_variable_init()

```
subroutine mo_mpr_startup::I0_variable_init (
    integer(i4), intent(in) iBasin )
```

level 0 variable initialization

following tasks are performed for L0 data sets

- cell id & numbering
- storage of cell coordinates (row and column id)
- empirical dist. of terrain slope
- flag to determine the presence of a particular soil id in this configuration of the model run If a variable is added or removed here, then it also has to be added or removed in the subroutine config_variables_set in module [mo_restart](#) and in the subroutine set_config in module [mo_set_netcdf_restart](#)

Parameters

in	integer(i4) :: iBasin	basin id
----	-----------------------	----------

Authors

Rohini Kumar

Date

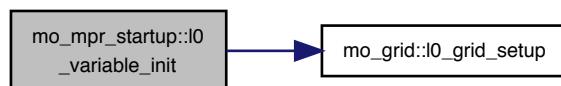
Jan 2013

by default; when iFlag_soilDB = 0

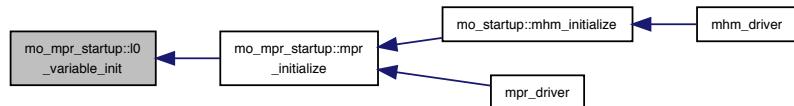
References [mo_mpr_global_variables::iflag_soildb](#), [mo_grid::I0_grid_setup\(\)](#), [mo_mpr_global_variables::I0_slope](#), [mo_mpr_global_variables::I0_slope_emp](#), [mo_mpr_global_variables::I0_soilid](#), [mo_common_variables::level0](#), [mo_mpr_global_variables::nsoilhorizons_mhm](#), [mo_mpr_global_variables::nsoiltypes](#), and [mo_mpr_global_variables::soildb](#).

Referenced by [mpr_initialize\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



15.46.2.4 mpr_initialize()

```
subroutine, public mo_mpr_startup::mpr_initialize ( )
```

Initialize main mHM variables.

Initialize main mHM variables for a given basin. Calls the following procedures in this order:

- Constant initialization.
- Generate soil database.
- Checking inconsistencies input fields.
- Variable initialization at level-0.
- Variable initialization at level-1.
- Variable initialization at level-11.
- Space allocation of remaining variable/parameters. Global variables will be used at this stage.

Authors

Luis Samaniego, Rohini Kumar

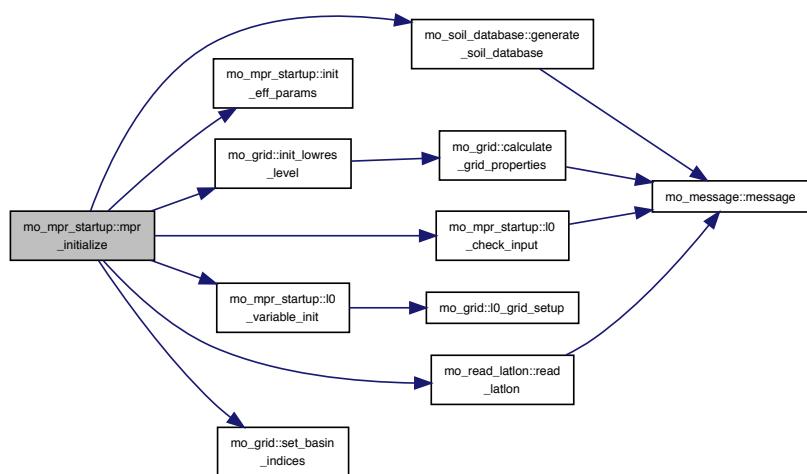
Date

Dec 2012

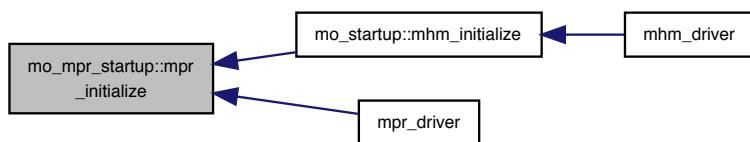
References `mo_soil_database::generate_soil_database()`, `mo_kind::i4`, `init_eff_params()`, `mo_grid::init_lowres_level()`, `mo_common_variables::l0_basin`, `l0_check_input()`, `mo_common_variables::l0_l1_remap`, `l0_variable_init()`, `mo_common_variables::level0`, `mo_common_variables::level1`, `mo_common_variables::nbasins`, `mo_read_latlon::read_latlon()`, `mo_common_variables::resolutionhydrology`, and `mo_grid::set_basin_indices()`.

Referenced by `mo_startup::mhm_initialize()`, and `mpo_driver()`.

Here is the call graph for this function:



Here is the caller graph for this function:



15.47 mo_mrm_constants Module Reference

Provides mRM specific constants.

Variables

- `integer(i4), parameter, public nouflxstate = 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 /)`

15.47.1 Detailed Description

Provides mRM specific constants.

Provides mRM specific constants such as flood plain elevation.

Authors

Stephan Thober

Date

Aug 2015

15.47.2 Variable Documentation

15.47.2.1 deltah

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

Referenced by `mo_mrm_net_startup::moveup()`.

15.47.2.2 given_ts

```
real(dp), dimension(19), parameter mo_mrm_constants::given_ts = (/ 60._dp, 120._dp, 180._dp, 240._dp, 300._dp, 360._dp, 600._dp, 720._dp, 900._dp, 1200._dp, 1800._dp, 3600._dp, 7200._dp, 10800._dp, 14400._dp, 21600._dp, 28800._dp, 43200._dp, 86400._dp /)
```

Referenced by `mo_mrm_init::mrm_init_param()`, and `mo_mrm_init::mrm_update_param()`.

15.47.2.3 maxnogauges

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

Referenced by `mo_mrm_read_config::mrm_read_config()`.

15.47.2.4 noutfluxstate

```
integer(i4), parameter, public mo_mrm_constants::noutfluxstate = 1_i4
```

15.47.2.5 nroutingstates

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

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

15.47.2.6 rout_space_weight

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

Referenced by mo_mrm_init::mrm_update_param().

15.48 mo_mrm_eval Module Reference

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

Functions/Subroutines

- subroutine, public [mrm_eval](#) (parameterset, runoff, sm_opti, basin_avg_tws, neutrons_opti, et_opti)
Runs mrm with a specific parameter set and returns required variables, e.g. runoff.

15.48.1 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

15.48.2 Function/Subroutine Documentation

15.48.2.1 mrm_eval()

```
subroutine, public mo_mrm_eval::mrm_eval (
    real(dp), dimension(:, ), intent(in) parameterset,
    real(dp), dimension(:, :, ), intent(out), optional, allocatable runoff,
    real(dp), dimension(:, :, ), intent(out), optional, allocatable sm_opti,
    real(dp), dimension(:, :, ), intent(out), optional, allocatable basin_avg_tws,
    real(dp), dimension(:, :, ), intent(out), optional, allocatable neutrons_opti,
    real(dp), dimension(:, :, ), intent(out), optional, allocatable et_opti )
```

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

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

Parameters

in	<i>real(dp), dimension(:) :: parameterset</i>	a set of global parameter (gamma) to run mHM, DIMENSION [no. of global_Parameters]
out	<i>real(dp), dimension(:, :) optional :: runoff</i>	returns runoff time series, DIMENSION [nTimeSteps, nGaugesTotal]
out	<i>real(dp), dimension(:, :) optional :: sm_opti</i>	dim1=ncells, dim2=time
out	<i>real(dp), dimension(:, :) optional :: basin_avg_tws</i>	dim1=time dim2=nBasins
out	<i>real(dp), dimension(:, :) optional :: neutrons_opti</i>	dim1=ncells, dim2=time
out	<i>real(dp), dimension(:, :) optional :: et_opti</i>	dim1=ncells, dim2=time

Authors

Stephan Thober

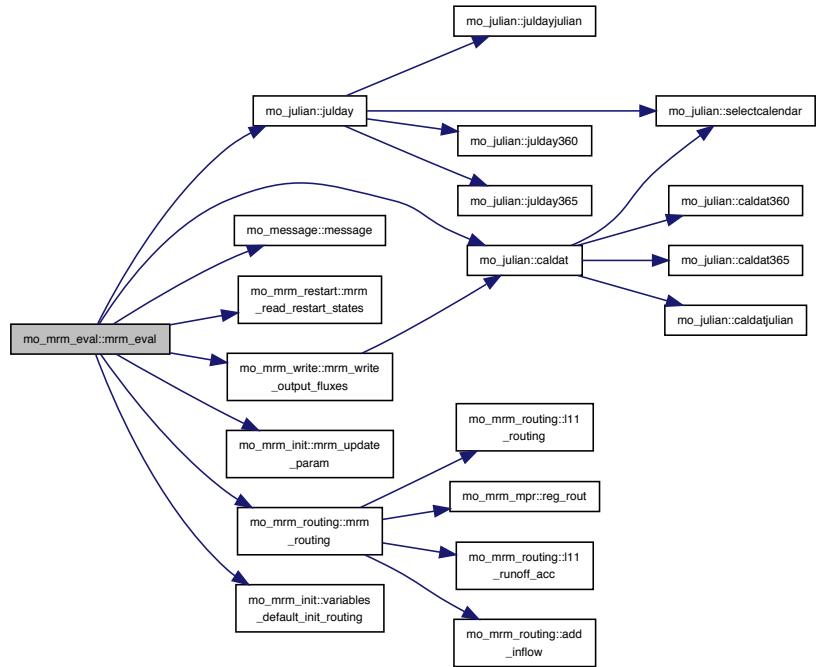
Date

Sep 2015

References mo_mrm_global_variables::basin_mrm, mo_julian::caldat(), mo_common_mhm_mrm_variables::dirrestartin, mo_kind::dp, mo_common_constants::hoursecs, mo_kind::i4, mo_mrm_global_variables::inflowgauge, mo_julian::julday(), mo_mrm_global_variables::l11_c1, mo_mrm_global_variables::l11_c2, mo_mrm_global_variables::l11_fromn, mo_mrm_global_variables::l11_l1_id, mo_mrm_global_variables::l11_length, mo_mrm_global_variables::l11_netperm, mo_mrm_global_variables::l11_nlinkfracfpimp, mo_mrm_global_variables::l11_noutlets, mo_mrm_global_variables::l11_qmod, mo_mrm_global_variables::l11_qout, mo_mrm_global_variables::l11_qtin, mo_mrm_global_variables::l11_qtr, mo_mrm_global_variables::l11_slope, mo_mrm_global_variables::l11_ton, mo_mrm_global_variables::l11_tsrou, mo_mrm_global_variables::l1_l11_id, mo_mrm_global_variables::l1_total_runoff_in, mo_common_mhm_mrm_variables::lcyearid, mo_common_variables::level1, mo_mrm_global_variables::level11, mo_message::message(), mo_mrm_restart::mrm_read_restart_states(), mo_mrm_routing::mrm_routing(), mo_mrm_global_variables::mrm_runoff, mo_mrm_init::mrm_update_param(), mo_mrm_write::mrm_write_output_fluxes(), mo_common_variables::nbasins, mo_common_mhm_mrm_variables::ntstepday, mo_common_mhm_mrm_variables::optimize, mo_mrm_global_variables::outputflxstate_mrm, mo_common_variables::processmatrix, mo_common_mhm_mrm_variables::read_restart, mo_common_variables::resolutionhydrology, mo_common_mhm_mrm_variables::resolutionrouting, mo_common_mhm_mrm_variables::simper, mo_common_mhm_mrm_variables::timestep, mo_mrm_global_variables::timestep_model_outputs_mrm, mo_mrm_init::variables_default_init_routing(), and mo_common_mhm_mrm_variables::warmingdays.

Referenced by mrm_driver().

Here is the call graph for this function:



Here is the caller graph for this function:



15.49 mo_mrm_file Module Reference

Provides file names and units for mRM.

Variables

- 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'

Unit for flow accumulation input data file.
 - integer, parameter **ufacc** = 56

flow direction input data file
 - character(len=*), parameter **file_fdir** = 'fdir.asc'

Unit for flow direction input data file.
 - integer, parameter **ufdir** = 57

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

15.49.1 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

15.49.2 Variable Documentation

15.49.2.1 file_config

character(len = *), parameter mo_mrm_file::file_config = 'ConfigFile.log'
file defining mHM's outputs

15.49.2.2 file_daily_discharge

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

file defining optimazation outputs

Referenced by mo_mrm_write::write_daily_obs_sim_discharge().

15.49.2.3 file_defoutput

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

file defining mRM's outputs

Referenced by mo_mrm_init::config_output(), mo_mrm_read_config::mrm_read_config(), and mo_mrm_init::print_startups_message().

15.49.2.4 file_facc

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

Referenced by mo_mrm_read_data::mrm_read_l0_data().

15.49.2.5 file_fdir

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

flow direction input data file

Referenced by mo_mrm_read_data::mrm_read_l0_data().

15.49.2.6 file_gaugeloc

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

gauge location input data file

Referenced by mo_mrm_read_data::mrm_read_l0_data().

15.49.2.7 file_main

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

Driver file.

Referenced by mo_mrm_init::print_startup_message().

15.49.2.8 file_mrm_output

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

file containing mrm output

Referenced by mo_mrm_write_fluxes_states::createoutputfile().

15.49.2.9 file_namelist_mrm

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

Namelist file name.

Referenced by mo_mrm_init::config_output(), and mrm_driver().

15.49.2.10 file_namelist_param_mrm

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

Parameter namelists file name.

Referenced by mo_mrm_init::config_output(), and mrm_driver().

15.49.2.11 ncf file_discharge

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

file defining optimazation outputs

Referenced by mo_mrm_write::write_daily_obs_sim_discharge().

15.49.2.12 uconfig

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

Unit for file defining mHM's outputs.

15.49.2.13 udaily_discharge

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

Unit for file optimazation outputs.

Referenced by `mo_mrm_write::write_daily_obs_sim_discharge()`.

15.49.2.14 udefoutput

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

Unit for file defining mRM's outputs.

Referenced by `mo_mrm_read_config::mrm_read_config()`.

15.49.2.15 udischarge

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

unit for discharge time series

Referenced by `mo_mrm_read_data::mrm_read_discharge()`.

15.49.2.16 ufacc

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

Unit for flow accumulation input data file.

Referenced by `mo_mrm_read_data::mrm_read_l0_data()`.

15.49.2.17 ufdir

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

Unit for flow direction input data file.

Referenced by `mo_mrm_read_data::mrm_read_l0_data()`.

15.49.2.18 ugaugeloc

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

Unit for gauge location input data file.

Referenced by `mo_mrm_read_data::mrm_read_l0_data()`.

15.49.2.19 unamelist_mrm

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

Unit for namelist.

Referenced by `mrm_driver()`.

15.49.2.20 unamelist_param_mrm

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

Unit for namelist.

Referenced by mrm_driver().

15.49.2.21 version

```
character(len = *), parameter mo_mrm_file::version = '1.1'
```

Current mHM model version.

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

15.49.2.22 version_date

```
character(len = *), parameter mo_mrm_file::version_date = 'Nov 2016'
```

Time of current mHM model version release.

Referenced by mo_mrm_init::print_startup_message().

15.50 mo_mrm_global_variables Module Reference

Global variables for mRM only.

Data Types

- type [basininfo_mrm](#)
- type [gaugingstation](#)

Variables

- logical [is_start](#)
- integer(i4) [timestep_model_outputs_mrm](#)
- logical, dimension(noutfluxstate) [outputfluxstate_mrm](#)
- character(256), dimension(:), allocatable, public [dirgauges](#)
- character(256), dimension(:), allocatable, public [dirtotalrunoff](#)
- character(256), public [filenametotalrunoff](#)
- character(256), public [varnametotalrunoff](#)
- 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 [nflowgaugestotal](#)
- 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 [l11_l1_id](#)
- real(dp), dimension(:, :), allocatable, public [l1_total_runoff_in](#)
- integer(i4), dimension(:), allocatable, public [l1_l11_id](#)
- integer(i4), dimension(:), allocatable, public [l11_fdir](#)
- integer(i4), dimension(:), allocatable, public [l11_noutlets](#)
- 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_tsroot](#)
- real(dp), dimension(:), allocatable, public [l11_c1](#)
- real(dp), dimension(:), allocatable, public [l11_c2](#)

15.50.1 Detailed Description

Global variables for mRM only.

TODO: add description

Authors

Luis Samaniego, Stephan Thober

Date

Aug 2015

15.50.2 Variable Documentation

15.50.2.1 basin_mrm

```
type(basininfo_mrm), dimension(:), allocatable, target, public mo_mrm_global_variables::basin_mrm
```

Referenced by mo_mrm_init::config_output(), mo_mrm_net_startup::l11_flow_direction(), mo_mrm_net_startup::l11_link_location(), mo_mrm_net_startup::l11_set_drain_outlet_gauges(), mo_mhm_eval::mhm_eval(), mo_mrm_eval::mrm_eval(), mo_mrm_init::mrm_init(), mo_mrm_init::mrm_init_param(), mo_mrm_read_config::mrm_read_config(), mo_mrm_read_data::mrm_read_l0_data(), mo_mrm_restart::mrm_read_restart_config(), mo_mrm_write::mrm_write(), mo_mrm_restart::mrm_write_restart(), mo_mrm_write::write_configfile(), and mo_mrm_write::write_daily_obs_sim_discharge().

15.50.2.2 dirgauges

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

Referenced by mo_mrm_init::config_output(), mo_mrm_read_config::mrm_read_config(), mo_write_ascii::write_configfile(), and mo_mrm_write::write_configfile().

15.50.2.3 dirtotalrunoff

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

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

15.50.2.4 filenametotalrunoff

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

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

15.50.2.5 gauge

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

Referenced by mo_mrm_objective_function_runoff::extract_runoff(), mo_mrm_read_config::mrm_read_config(), mo_mrm_read_data::mrm_read_discharge(), mo_mrm_write::mrm_write(), mo_mrm_objective_function_runoff::multi_objective_ae_fdc_lsv_nse_djf(), mo_write_ascii::write_configfile(), mo_mrm_write::write_configfile(), and mo_mrm_write::write_daily_obs_sim_discharge().

15.50.2.6 inflowgauge

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

Referenced by `mo_mhm_eval::mhm_eval()`, `mo_mrm_eval::mrm_eval()`, `mo_mrm_read_config::mrm_read_config()`, `mo_mrm_read_data::mrm_read_discharge()`, `mo_write_ascii::write_configfile()`, and `mo_mrm_write::write_configfile()`.

15.50.2.7 `is_start`

```
logical mo_mrm_global_variables::is_start
```

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

15.50.2.8 `l0_dracell`

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

Referenced by `mo_mrm_net_startup::l11_set_drain_outlet_gauges()`.

15.50.2.9 `l0_drasc`

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

Referenced by `mo_mrm_net_startup::l11_flow_direction()`, `mo_mrm_net_startup::l11_link_location()`, and `mo_mrm_net_startup::l11_set_drain_outlet_gauges()`.

15.50.2.10 `l0_facc`

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

Referenced by `mo_mrm_init::l0_check_input_routing()`, `mo_mrm_net_startup::l11_flow_direction()`, and `mo_mrm_read_data::mrm_read_l0_data()`.

15.50.2.11 `l0_fdir`

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

Referenced by `mo_mrm_init::l0_check_input_routing()`, `mo_mrm_net_startup::l11_flow_direction()`, `mo_mrm_net_startup::l11_link_location()`, `mo_mrm_net_startup::l11_set_drain_outlet_gauges()`, `mo_mrm_net_startup::l11_stream_features()`, and `mo_mrm_read_data::mrm_read_l0_data()`.

15.50.2.12 `l0_floodplain`

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

Referenced by `mo_mrm_net_startup::l11_fraction_sealed_floodplain()`, and `mo_mrm_net_startup::l11_stream_features()`.

15.50.2.13 l0_gaugeloc

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

Referenced by mo_mrm_net_startup::l11_set_drain_outlet_gauges(), and mo_mrm_read_data::mrm_read_l0_data().

15.50.2.14 l0_inflowgaugeloc

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

Referenced by mo_mrm_net_startup::l11_set_drain_outlet_gauges(), and mo_mrm_read_data::mrm_read_l0_data().

15.50.2.15 l0_l11_remap

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

Referenced by mo_mrm_net_startup::l11_flow_direction(), mo_mrm_net_startup::l11_set_drain_outlet_gauges(), and mo_mrm_init::mrm_init().

15.50.2.16 l0_streamnet

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

Referenced by mo_mrm_net_startup::l11_stream_features().

15.50.2.17 l11_afloodplain

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

Referenced by mo_mrm_net_startup::l11_fraction_sealed_floodplain(), mo_mrm_net_startup::l11_stream_features(), mo_mrm_restart::mrm_read_restart_config(), and mo_mrm_restart::mrm_write_restart().

15.50.2.18 l11_c1

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

Referenced by mo_mhm_eval::mhm_eval(), mo_mrm_eval::mrm_eval(), mo_mrm_restart::mrm_read_restart_states(), mo_mrm_init::mrm_update_param(), mo_mrm_restart::mrm_write_restart(), mo_mrm_init::variables_alloc_routing(), and mo_mrm_init::variables_default_init_routing().

15.50.2.19 l11_c2

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

Referenced by mo_mhm_eval::mhm_eval(), mo_mrm_eval::mrm_eval(), mo_mrm_restart::mrm_read_restart_states(), mo_mrm_init::mrm_update_param(), mo_mrm_restart::mrm_write_restart(), mo_mrm_init::variables_alloc_routing(), and mo_mrm_init::variables_default_init_routing().

15.50.2.20 l11_colout

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

Referenced by mo_mrm_net_startup::l11_flow_direction(), mo_mrm_net_startup::l11_link_location(), mo_mrm_restart::mrm_read_restart_config(), and mo_mrm_restart::mrm_write_restart().

15.50.2.21 l11_fcol

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

Referenced by mo_mrm_net_startup::l11_link_location(), mo_mrm_net_startup::l11_stream_features(), mo_mrm_restart::mrm_read_restart_config(), and mo_mrm_restart::mrm_write_restart().

15.50.2.22 l11_fdir

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

Referenced by mo_mrm_net_startup::l11_flow_direction(), mo_mrm_net_startup::l11_routing_order(), mo_mrm_net_startup::l11_set_network_topology(), mo_mrm_restart::mrm_read_restart_config(), and mo_mrm_restart::mrm_write_restart().

15.50.2.23 l11_fromn

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

Referenced by mo_mrm_net_startup::l11_link_location(), mo_mrm_net_startup::l11_routing_order(), mo_mrm_net_startup::l11_set_network_topology(), mo_mhm_eval::mhm_eval(), mo_mrm_eval::mrm_eval(), mo_mrm_restart::mrm_read_restart_config(), mo_mrm_restart::mrm_write_restart(), mo_write_ascii::write_configfile(), and mo_mrm_write::write_configfile().

15.50.2.24 l11_frow

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

Referenced by mo_mrm_net_startup::l11_link_location(), mo_mrm_net_startup::l11_stream_features(), mo_mrm_restart::mrm_read_restart_config(), and mo_mrm_restart::mrm_write_restart().

15.50.2.25 l11_k

```
real(dp), dimension(:), allocatable, public mo_mrm_global_variables::l11_k
```

Referenced by mo_mrm_restart::mrm_read_restart_states(), mo_mrm_restart::mrm_write_restart(), mo_mrm_init::variables_alloc_routing(), and mo_mrm_init::variables_default_init_routing().

15.50.2.26 l11_l1_id

```
integer(i4), dimension(:), allocatable, public mo_mrm_global_variables::l11_l1_id
```

Referenced by `mo_mrm_net_startup::l11_l1_mapping()`, `mo_mhm_eval::mhm_eval()`, `mo_mrm_eval::mrm_eval()`, `mo_mrm_restart::mrm_read_restart_config()`, `mo_mrm_restart::mrm_write_restart()`, and `mo_mrm_write::write_configfile()`.

15.50.2.27 l11_label

```
integer(i4), dimension(:), allocatable, public mo_mrm_global_variables::l11_label
```

Referenced by `mo_mrm_net_startup::l11_routing_order()`, `mo_mrm_restart::mrm_read_restart_config()`, `mo_mrm_restart::mrm_write_restart()`, `mo_write_ascii::write_configfile()`, and `mo_mrm_write::write_configfile()`.

15.50.2.28 l11_length

```
real(dp), dimension(:), allocatable, public mo_mrm_global_variables::l11_length
```

Referenced by `mo_mrm_net_startup::l11_stream_features()`, `mo_mhm_eval::mhm_eval()`, `mo_mrm_eval::mrm_eval()`, `mo_mrm_restart::mrm_read_restart_config()`, `mo_mrm_restart::mrm_write_restart()`, `mo_write_ascii::write_configfile()`, and `mo_mrm_write::write_configfile()`.

15.50.2.29 l11_netperm

```
integer(i4), dimension(:), allocatable, public mo_mrm_global_variables::l11_netperm
```

Referenced by `mo_mrm_net_startup::l11_link_location()`, `mo_mrm_net_startup::l11_routing_order()`, `mo_mrm_net_startup::l11_stream_features()`, `mo_mhm_eval::mhm_eval()`, `mo_mrm_eval::mrm_eval()`, `mo_mrm_restart::mrm_read_restart_config()`, `mo_mrm_restart::mrm_write_restart()`, `mo_write_ascii::write_configfile()`, and `mo_mrm_write::write_configfile()`.

15.50.2.30 l11_nlinkfracfpimp

```
real(dp), dimension(:, :), allocatable, public mo_mrm_global_variables::l11_nlinkfracfpimp
```

Referenced by `mo_mrm_net_startup::l11_fraction_sealed_floodplain()`, `mo_mhm_eval::mhm_eval()`, `mo_mrm_eval::mrm_eval()`, `mo_mrm_restart::mrm_read_restart_states()`, and `mo_mrm_restart::mrm_write_restart()`.

15.50.2.31 l11_noutlets

```
integer(i4), dimension(:), allocatable, public mo_mrm_global_variables::l11_noutlets
```

Referenced by `mo_mrm_net_startup::l11_flow_direction()`, `mo_mrm_net_startup::l11_fraction_sealed_floodplain()`, `mo_mrm_net_startup::l11_link_location()`, `mo_mrm_net_startup::l11_routing_order()`, `mo_mrm_net_startup::l11_stream_features()`, `mo_mhm_eval::mhm_eval()`, `mo_mrm_eval::mrm_eval()`, and `mo_mrm_restart::mrm_read_restart_config()`.

15.50.2.32 l11_qmod

```
real(dp), dimension(:), allocatable, public mo_mrm_global_variables::l11_qmod
Referenced by mo_mhm_eval::mhm_eval(), mo_mrm_eval::mrm_eval(), mo_mrm_restart::mrm_read_restart_
_states(), mo_mrm_restart::mrm_write_restart(), mo_mrm_init::variables_alloc_routing(), and mo_mrm_init_
::variables_default_init_routing().
```

15.50.2.33 l11_qout

```
real(dp), dimension(:), allocatable, public mo_mrm_global_variables::l11_qout
Referenced by mo_mhm_eval::mhm_eval(), mo_mrm_eval::mrm_eval(), mo_mrm_restart::mrm_read_restart_
_states(), mo_mrm_restart::mrm_write_restart(), mo_mrm_init::variables_alloc_routing(), and mo_mrm_init_
::variables_default_init_routing().
```

15.50.2.34 l11_qtin

```
real(dp), dimension(:, :), allocatable, public mo_mrm_global_variables::l11_qtin
Referenced by mo_mhm_eval::mhm_eval(), mo_mrm_eval::mrm_eval(), mo_mrm_restart::mrm_read_restart_
_states(), mo_mrm_restart::mrm_write_restart(), mo_mrm_init::variables_alloc_routing(), and mo_mrm_init_
::variables_default_init_routing().
```

15.50.2.35 l11_qtr

```
real(dp), dimension(:, :), allocatable, public mo_mrm_global_variables::l11_qtr
Referenced by mo_mhm_eval::mhm_eval(), mo_mrm_eval::mrm_eval(), mo_mrm_restart::mrm_read_restart_
_states(), mo_mrm_restart::mrm_write_restart(), mo_mrm_init::variables_alloc_routing(), and mo_mrm_init_
::variables_default_init_routing().
```

15.50.2.36 l11_rorder

```
integer(i4), dimension(:), allocatable, public mo_mrm_global_variables::l11_rorder
Referenced by mo_mrm_net_startup::l11_routing_order(), mo_mrm_restart::mrm_read_restart_config(), mo_-
mrm_restart::mrm_write_restart(), mo_write_ascii::write_configfile(), and mo_mrm_write::write_configfile().
```

15.50.2.37 l11_rowout

```
integer(i4), dimension(:), allocatable, public mo_mrm_global_variables::l11_rowout
Referenced by mo_mrm_net_startup::l11_flow_direction(), mo_mrm_net_startup::l11_link_location(), mo_mrm_-
restart::mrm_read_restart_config(), and mo_mrm_restart::mrm_write_restart().
```

15.50.2.38 l11_sink

```
logical, dimension(:), allocatable, public mo_mrm_global_variables::l11_sink
```

Referenced by mo_mrm_net_startup::l11_routing_order(), mo_mrm_restart::mrm_read_restart_config(), and mo_mrm_restart::mrm_write_restart().

15.50.2.39 l11_slope

```
real(dp), dimension(:), allocatable, public mo_mrm_global_variables::l11_slope
```

Referenced by mo_mrm_net_startup::l11_stream_features(), mo_mhm_eval::mhm_eval(), mo_mrm_eval::mrm_eval(), mo_mrm_restart::mrm_read_restart_config(), mo_mrm_restart::mrm_write_restart(), mo_write_ascii::write_configfile(), and mo_mrm_write::write_configfile().

15.50.2.40 l11_tcol

```
integer(i4), dimension(:), allocatable, public mo_mrm_global_variables::l11_tcol
```

Referenced by mo_mrm_net_startup::l11_link_location(), mo_mrm_net_startup::l11_stream_features(), mo_mrm_restart::mrm_read_restart_config(), and mo_mrm_restart::mrm_write_restart().

15.50.2.41 l11_ton

```
integer(i4), dimension(:), allocatable, public mo_mrm_global_variables::l11_ton
```

Referenced by mo_mrm_net_startup::l11_routing_order(), mo_mrm_net_startup::l11_set_network_topology(), mo_mhm_eval::mhm_eval(), mo_mrm_eval::mrm_eval(), mo_mrm_restart::mrm_read_restart_config(), mo_mrm_restart::mrm_write_restart(), mo_write_ascii::write_configfile(), and mo_mrm_write::write_configfile().

15.50.2.42 l11_trow

```
integer(i4), dimension(:), allocatable, public mo_mrm_global_variables::l11_trow
```

Referenced by mo_mrm_net_startup::l11_link_location(), mo_mrm_net_startup::l11_stream_features(), mo_mrm_restart::mrm_read_restart_config(), and mo_mrm_restart::mrm_write_restart().

15.50.2.43 l11_tsroute

```
real(dp), dimension(:), allocatable, public mo_mrm_global_variables::l11_tsroute
```

Referenced by mo_mhm_eval::mhm_eval(), mo_mrm_eval::mrm_eval(), mo_mrm_init::mrm_init_param(), mo_mrm_restart::mrm_read_restart_config(), mo_mrm_init::mrm_update_param(), and mo_mrm_restart::mrm_write_restart().

15.50.2.44 l11_xi

```
real(dp), dimension(:), allocatable, public mo_mrm_global_variables::l11_xi
```

Referenced by `mo_mrm_restart::mrm_read_restart_states()`, `mo_mrm_restart::mrm_write_restart()`, `mo_mrm_init::variables_alloc_routing()`, and `mo_mrm_init::variables_default_init_routing()`.

15.50.2.45 l1_l11_id

```
integer(i4), dimension(:), allocatable, public mo_mrm_global_variables::l1_l11_id
```

Referenced by `mo_mrm_net_startup::l11_l1_mapping()`, `mo_mhm_eval::mhm_eval()`, `mo_mrm_eval::mrm_eval()`, `mo_mrm_restart::mrm_read_restart_config()`, `mo_mrm_restart::mrm_write_restart()`, `mo_write_ascii::write_configfile()`, and `mo_mrm_write::write_configfile()`.

15.50.2.46 l1_l11_remap

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

Referenced by `mo_mrm_init::mrm_init()`.

15.50.2.47 l1_total_runoff_in

```
real(dp), dimension(:, :), allocatable, public mo_mrm_global_variables::l1_total_runoff_in
```

Referenced by `mo_mrm_eval::mrm_eval()`, and `mo_mrm_read_data::mrm_read_total_runoff()`.

15.50.2.48 level11

```
type(grid), dimension(:), allocatable, target, public mo_mrm_global_variables::level11
```

Referenced by `mo_mrm_write_fluxes_states::createoutputfile()`, `mo_mrm_net_startup::l11_flow_direction()`, `mo_mrm_net_startup::l11_fraction_sealed_floodplain()`, `mo_mrm_net_startup::l11_l1_mapping()`, `mo_mrm_net_startup::l11_link_location()`, `mo_mrm_net_startup::l11_routing_order()`, `mo_mrm_net_startup::l11_set_network_topology()`, `mo_mrm_net_startup::l11_stream_features()`, `mo_mhm_eval::mhm_eval()`, `mo_mrm_eval::mrm_eval()`, `mo_mrm_init::mrm_init()`, `mo_mrm_restart::mrm_read_restart_config()`, `mo_mrm_restart::mrm_read_restart_states()`, `mo_mrm_init::mrm_update_param()`, `mo_mrm_restart::mrm_write_restart()`, `mo_mrm_init::variables_alloc_routing()`, `mo_write_ascii::write_configfile()`, and `mo_mrm_write::write_configfile()`.

15.50.2.49 mrm_runoff

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

Referenced by `mo_mhm_eval::mhm_eval()`, `mo_mrm_eval::mrm_eval()`, `mo_mrm_read_data::mrm_read_discharge()`, and `mo_mrm_write::mrm_write()`.

15.50.2.50 ngaugestotal

```
integer(i4), public mo_mrm_global_variables::ngaugestotal
```

Referenced by `mo_mrm_read_config::mrm_read_config()`, `mo_mrm_read_data::mrm_read_discharge()`, `mo_mrm_write::mrm_write()`, `mo_write_ascii::write_configfile()`, and `mo_mrm_write::write_configfile()`.

15.50.2.51 ninflowgaugestotal

```
integer(i4), public mo_mrm_global_variables::ninflowgaugestotal
```

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

15.50.2.52 nmeasperday

```
integer(i4), public mo_mrm_global_variables::nmeasperday
```

Referenced by mo_mrm_objective_function_runoff::extract_runoff(), mo_mrm_read_data::mrm_read_discharge(), and mo_mrm_objective_function_runoff::multi_objective_ae_fdc_lsv_nse_djf().

15.50.2.53 outputflxstate_mrm

```
logical, dimension(noutflxstate) mo_mrm_global_variables::outputflxstate_mrm
```

Referenced by mo_mhm_eval::mhm_eval(), mo_mrm_eval::mrm_eval(), mo_mrm_read_config::mrm_read_config(), mo_mrm_write_fluxes_states::newoutputdataset(), and mo_mrm_write_fluxes_states::updatedataset().

15.50.2.54 timestep_model_outputs_mrm

```
integer(i4) mo_mrm_global_variables::timestep_model_outputs_mrm
```

Referenced by mo_mhm_eval::mhm_eval(), mo_mrm_eval::mrm_eval(), and mo_mrm_read_config::mrm_read_config().

15.50.2.55 varnametotalrunoff

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

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

15.51 mo_mrm_init Module Reference

Wrapper for initializing Routing.

Functions/Subroutines

- subroutine, public **mrm_init** (file_namelist, unamelist, file_namelist_param, unamelist_param)

Initialize all mRM variables at all levels (i.e., L0, L1, and L11).
- subroutine **print_startup_message** (file_namelist, file_namelist_param)

TODO: add description.
- subroutine **config_output**

TODO: add description.

- subroutine, public `variables_default_init_routing`
Default initialization mRM related L11 variables.
- subroutine `l0_check_input_routing` (L0Basin_iBasin)
TODO: add description.
- subroutine `variables_alloc_routing` (iBasin)
TODO: add description.
- subroutine `mrm_init_param` (iBasin, param)
TODO: add description.
- subroutine, public `mrm_update_param` (iBasin, param)
TODO: add description.

15.51.1 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

15.51.2 Function/Subroutine Documentation

15.51.2.1 config_output()

```
subroutine mo_mrm_init::config_output ( )
```

TODO: add description.

TODO: add description

Authors

Robert Scheppe

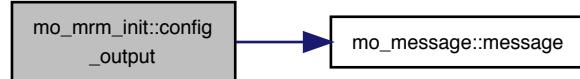
Date

Jun 2018

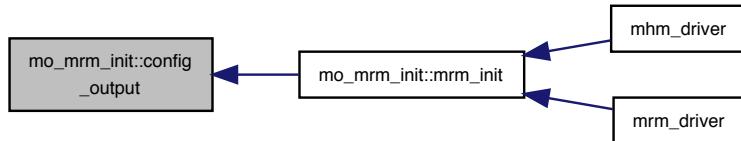
References `mo_mrm_global_variables::basin_mrm`, `mo_mrm_global_variables::dirgauges`, `mo_common_variables::dirlcover`, `mo_common_variables::dirmorpho`, `mo_common_variables::dirout`, `mo_mrm_file::file::defoutput`, `mo_mrm_file::file_namelist_mrm`, `mo_mrm_file::file_namelist_param_mrm`, `mo_kind::i4`, `mo_message::message()`, and `mo_common_variables::nbasins`.

Referenced by `mrm_init()`.

Here is the call graph for this function:



Here is the caller graph for this function:



15.51.2.2 l0_check_input_routing()

```
subroutine mo_mrm_init::l0_check_input_routing (
    integer(i4), intent(in) L0Basin_iBasin )
```

TODO: add description.

TODO: add description

Parameters

in	integer(i4) :: L0Basin_iBasin	
----	-------------------------------	--

Authors

Robert Schwepppe

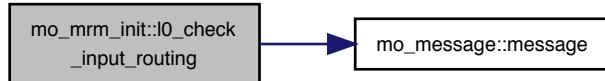
Date

Jun 2018

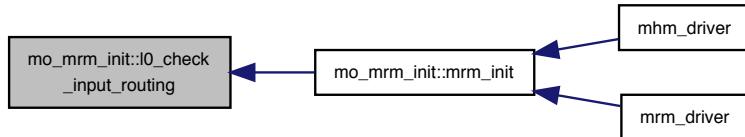
References mo_kind::i4, mo_mrm_global_variables::l0_facc, mo_mrm_global_variables::l0_fdir, mo_common_variables::level0, mo_message::message(), mo_message::message_text, and mo_common_constants::nodata_i4.

Referenced by mrm_init().

Here is the call graph for this function:



Here is the caller graph for this function:



15.51.2.3 mrm_init()

```

subroutine, public mo_mrm_init::mrm_init (
    character(*), intent(in) file_namelist,
    integer, intent(in) unamelist,
    character(*), intent(in) file_namelist_param,
    integer, intent(in) unamelist_param )

```

Initialize all mRM variables at all levels (i.e., L0, L1, and L11).

Initialize all mRM variables at all levels (i.e., L0, L1, and L11) either with default values or with values from restart file. The L0 mask (L0_mask), L0 elevation (L0_elev), and L0 land cover (L0_LCover) can be provided as optional variables to save memory because these variable will then not be read in again.

Parameters

in	character(*) :: file_namelist, file_namelist_param	
in	integer :: unamelist, unamelist_param	
in	character(*) :: file_namelist, file_namelist_param	
in	integer :: unamelist, unamelist_param	

Authors

Stephan Thober

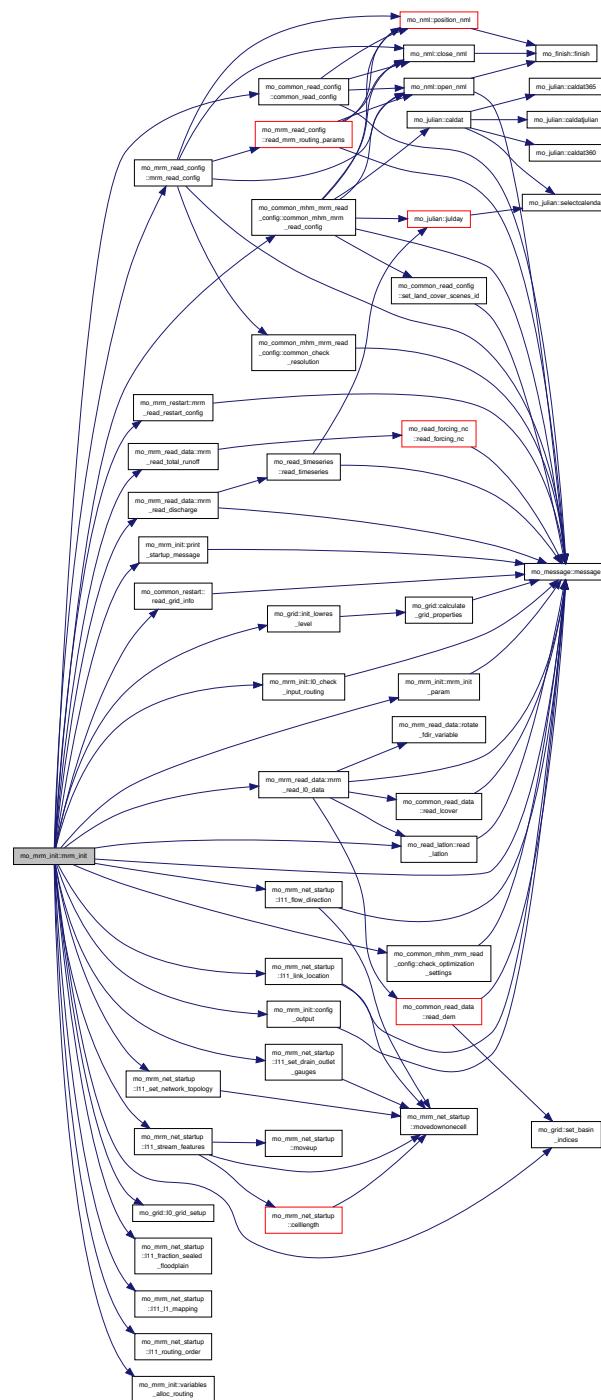
Date

Aug 2015

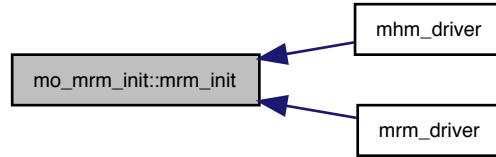
References mo_mrm_global_variables::basin_mrm, mo_common_mhm_mrm_read_config::check_optimization_settings(), mo_common_mhm_mrm_read_config::common_mhm_mrm_read_config(), mo_common_read_config::common_read_config(), config_output(), mo_common_mhm_mrm_variables::dirrestartin, mo_common_variables::global_parameters, mo_kind::i4, mo_grid::init_lowres_level(), mo_common_variables::l0_basin, l0_check_input_routing(), mo_grid::l0_grid_setup(), mo_mrm_global_variables::l0_l11_remap, mo_common_variables::l0_l11_remap, mo_mrm_net_startup::l11_flow_direction(), mo_mrm_net_startup::l11_fraction_sealed_floodplain(), mo_mrm_net_startup::l11_l1_mapping(), mo_mrm_net_startup::l11_link_location(), mo_mrm_net_startup::l11_routing_order(), mo_mrm_net_startup::l11_set_drain_outlet_gauges(), mo_mrm_net_startup::l11_set_network_topology(), mo_mrm_net_startup::l11_stream_features(), mo_mrm_global_variables::l1_l11_remap, mo_common_variables::level0, mo_common_variables::level1, mo_mrm_global_variables::level11, mo_message::message(), mo_common_mhm_mrm_variables::mrm_coupling_mode, 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_read_data::mrm_read_total_runoff(), mo_common_variables::nbasins, mo_common_constants::nodata_dp, mo_common_constants::nodata_i4, print_startup_message(), mo_common_variables::processmatrix, mo_common_restart::read_grid_info(), mo_read_latlon::read_latlon(), mo_common_mhm_mrm_variables::read_restart, mo_common_variables::resolutionhydrology, mo_common_mhm_mrm_variables::resolutionrouting, mo_grid::set_basin_indices(), and variables_alloc_routing().

Referenced by mhm_driver(), and mrm_driver().

Here is the call graph for this function:



Here is the caller graph for this function:



15.51.2.4 mrm_init_param()

```
subroutine mo_mrm_init::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

Robert Scheppele

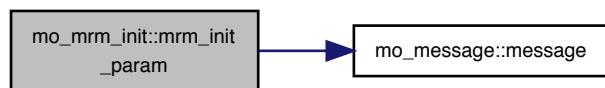
Date

Jun 2018

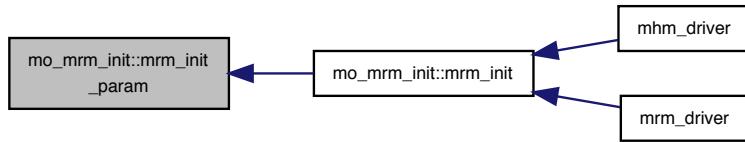
References mo_mrm_global_variables::basin_mrm, mo_kind::dp, mo_mrm_constants::given_ts, mo_common_constants::hoursecs, mo_kind::i4, mo_common_variables::iflag_coordinate_sys, mo_mrm_global_variables::i11_tsroute, mo_message::message(), mo_common_variables::nbasins, mo_common_variables::processmatrix, mo_common_mhm_mrm_variables::resolutionrouting, and mo_common_mhm_mrm_variables::timestep.

Referenced by mrm_init().

Here is the call graph for this function:



Here is the caller graph for this function:



15.51.2.5 mrm_update_param()

```

subroutine, public mo_mrm_init::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

Robert Schweppel

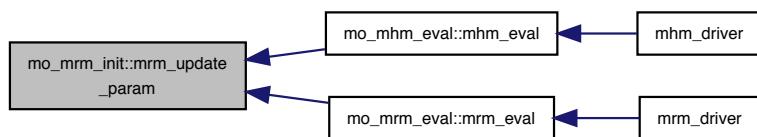
Date

Jun 2018

References mo_kind::dp, mo_mrm_constants::given_ts, mo_kind::i4, mo_common_variables::iflag_cordinate_sys, mo_mrm_global_variables::l11_c1, mo_mrm_global_variables::l11_c2, mo_mrm_global_variables::l11_tsroute, mo_mrm_global_variables::level11, mo_common_mhm_mrm_variables::resolutionrouting, and mo_mrm_constants::rout_space_weight.

Referenced by mo_mhm_eval::mhm_eval(), and mo_mrm_eval::mrm_eval().

Here is the caller graph for this function:



15.51.2.6 print_startup_message()

```
subroutine mo_mrm_init::print_startup_message (
    character(*), intent(in) file_namelist,
    character(*), intent(in) file_namelist_param )
```

TODO: add description.

TODO: add description

Parameters

in	character(*) :: file_namelist, file_namelist_param	
in	character(*) :: file_namelist, file_namelist_param	

Authors

Robert Schwepppe

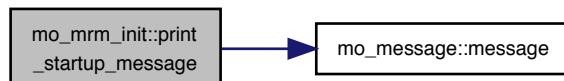
Date

Jun 2018

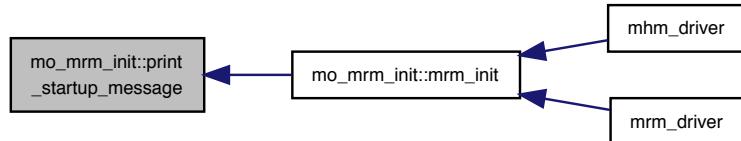
References mo_mrm_file::file_defoutput, mo_mrm_file::file_main, mo_kind::i4, mo_message::message(), mo_message::message_text, mo_string_utils::separator, mo_mrm_file::version, and mo_mrm_file::version_date.

Referenced by mrm_init().

Here is the call graph for this function:



Here is the caller graph for this function:



15.51.2.7 variables_alloc_routing()

```
subroutine mo_mrm_init::variables_alloc_routing (
    integer(i4), intent(in) iBasin )
```

TODO: add description.

TODO: add description

Parameters

in	integer(i4) :: iBasin	
----	-----------------------	--

Authors

Robert Schwepppe

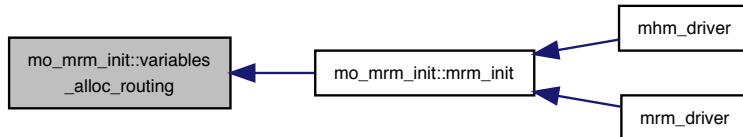
Date

Jun 2018

References mo_kind::dp, mo_kind::i4, mo_mrm_global_variables::l11_c1, mo_mrm_global_variables::l11_c2, mo_mrm_global_variables::l11_k, mo_mrm_global_variables::l11_qmod, mo_mrm_global_variables::l11_qout, mo_mrm_global_variables::l11_qtin, mo_mrm_global_variables::l11_qtr, mo_mrm_global_variables::l11_xi, mo_mrm_global_variables::level11, and mo_mrm_constants::nroutingstates.

Referenced by mrm_init().

Here is the caller graph for this function:



15.51.2.8 variables_default_init_routing()

```
subroutine, public mo_mrm_init::variables_default_init_routing ( )
```

Default initialization mRM related L11 variables.

Default initialization of mHM related L11 variables (e.g., states, fluxes, and parameters) as per given constant values given in [mo_mhm_constants](#). Variables initialized here is defined in the [mo_global_variables.f90](#) file. Only Variables that are defined in the variables_alloc subroutine are initialized 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_ncdf_restart](#). ADDITIONAL INFORMATION variables_default_init_routing call [variables_default_init\(\)](#)

Authors

Stephan Thober, Rohini Kumar, and Juliane Mai

Date

Aug 2015

Authors

Robert Schweißgeher

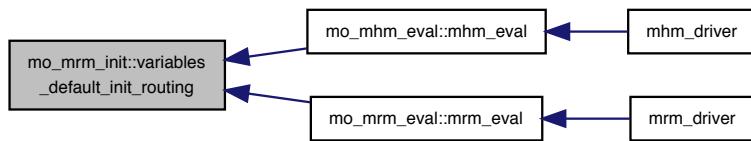
Date

Jun 2018

References `mo_mrm_global_variables::l11_c1`, `mo_mrm_global_variables::l11_c2`, `mo_mrm_global_variables::l11_k`, `mo_mrm_global_variables::l11_qmod`, `mo_mrm_global_variables::l11_qout`, `mo_mrm_global_variables::l11_qtin`, `mo_mrm_global_variables::l11_qtr`, `mo_mrm_global_variables::l11_xi`, and `mo_common_constants::p1_initstatefluxes`.

Referenced by `mo_mhm_eval::mhm_eval()`, and `mo_mrm_eval::mrm_eval()`.

Here is the caller graph for this function:



15.52 mo_mrm_mpr Module Reference

Perform Multiscale Parameter Regionalization on Routing Parameters.

Functions/Subroutines

- subroutine, public **reg_rout** (param, length, slope, fFPimp, TS, C1, C2)
Regionalized routing.

15.52.1 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

15.52.2 Function/Subroutine Documentation

15.52.2.1 reg_rout()

```
subroutine, public mo_mrm_mpr::reg_rout (
    real(dp), dimension(5), intent(in) param,
    real(dp), dimension(:, ), intent(in) length,
    real(dp), dimension(:, ), intent(in) slope,
    real(dp), dimension(:, ), intent(in) fFPimp,
    real(dp), intent(in) TS,
    real(dp), dimension(:, ), intent(out) C1,
    real(dp), dimension(:, ), intent(out) C2 )
```

Regionalized routing.

sets up the Regionalized Routing parameters Global parameters needed (see mhm_parameter.nml):

- param(1) = muskingumTravelTime_constant
- param(2) = muskingumTravelTime_riverLength
- param(3) = muskingumTravelTime_riverSlope
- param(4) = muskingumTravelTime_impervious
- param(5) = muskingumAttenuation_riverSlope

Parameters

in	<i>real(dp), dimension(5) :: param</i>	input parameter
in	<i>real(dp), dimension(:,) :: length</i>	[m] total length
in	<i>real(dp), dimension(:,) :: slope</i>	average slope
in	<i>real(dp), dimension(:,) :: fFPimp</i>	fraction of the flood plain with impervious layer
in	<i>real(dp) :: TS</i>	- [h] time step in
out	<i>real(dp), dimension(:,) :: C1</i>	routing parameter C1 (Chow, 25-41)
out	<i>real(dp), dimension(:,) :: C2</i>	routing parameter C2 ("")

Authors

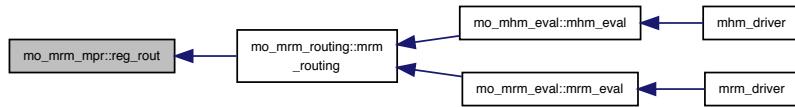
Stephan Thober, Rohini Kumar

Date

Dec 2012

Referenced by mo_mrm_routing::mrm_routing().

Here is the caller graph for this function:



15.53 mo_mrm_net_startup Module Reference

Startup drainage network for mHM.

Functions/Subroutines

- subroutine, public [l11_l1_mapping](#) (iBasin)

TODO: add description.
- subroutine, public [l11_flow_direction](#) (iBasin)

Determine the flow direction of the upscaled river network at level L11.
- subroutine, public [l11_set_network_topology](#) (iBasin)

Set network topology.
- subroutine, public [l11_routing_order](#) (iBasin)

Find routing order, headwater cells and sink.
- subroutine, public [l11_link_location](#) (iBasin)

Estimate the LO (row,col) location for each routing link at level L11.
- subroutine, public [l11_set_drain_outlet_gauges](#) (iBasin)

Draining cell identification and Set gauging node.
- subroutine, public [l11_stream_features](#) (iBasin)

Stream features (stream network and floodplain)
- subroutine, public [l11_fraction_sealed_floodplain](#) (LCClassImp, do_init)

Fraction of the flood plain with impervious cover.
- subroutine [moveup](#) (elev0, fDir0, fi, fj, ss, nn)

TODO: add description.
- subroutine [movedownonecell](#) (fDir, iRow, jCol)

TODO: add description.
- subroutine [celllength](#) (iBasin, fDir, iRow, jCol, iCoorSystem, length)

TODO: add description.
- subroutine, public [get_distance_two_lat_lon_points](#) (lat1, long1, lat2, long2, distance_out)

estimate distance in [m] between two points in a lat-lon

15.53.1 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

15.53.2 Function/Subroutine Documentation

15.53.2.1 celllength()

```
subroutine mo_mrm_net_startup::celllength (
    integer(i4), intent(in) iBasin,
    integer(i4), intent(in) fDir,
    integer(i4), intent(in) iRow,
    integer(i4), intent(in) jCol,
    integer(i4), intent(in) iCoorSystem,
    real(dp), intent(out) length )
```

TODO: add description.

TODO: add description

Parameters

in	<i>integer(i4) :: iBasin</i>	
in	<i>integer(i4) :: fDir</i>	
in	<i>integer(i4) :: iRow</i>	
in	<i>integer(i4) :: jCol</i>	
in	<i>integer(i4) :: iCoorSystem</i>	
out	<i>real(dp) :: length</i>	

Authors

Robert Schweppe

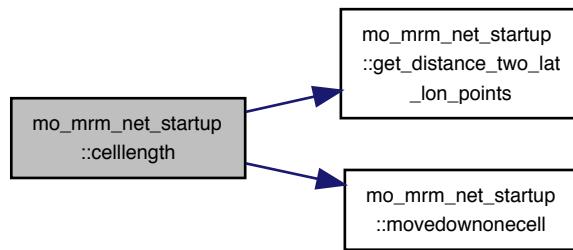
Date

Jun 2018

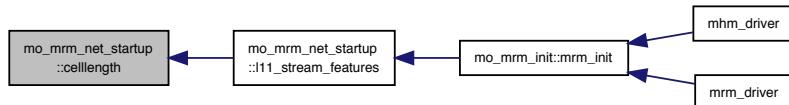
References `get_distance_two_lat_lon_points()`, `mo_common_variables::l0_basin`, `mo_common_variables::level0`, `movedownonecell()`, and `mo_constants::sqrt2_dp`.

Referenced by `l11_stream_features()`.

Here is the call graph for this function:



Here is the caller graph for this function:



15.53.2.2 `get_distance_two_lat_lon_points()`

```

subroutine, public mo_mrm_net_startup::get_distance_two_lat_lon_points (
    real(dp), intent(in) lat1,
    real(dp), intent(in) long1,
    real(dp), intent(in) lat2,
    real(dp), intent(in) long2,
    real(dp), intent(out) distance_out )
  
```

estimate distance in [m] between two points in a lat-lon

estimate distance in [m] between two points in a lat-lon Code is based on one that is implemented in the VIC-3L model

Parameters

in	<code>real(dp) :: lat1, long1, lat2, long2</code>	latitude of point-1
in	<code>real(dp) :: lat1, long1, lat2, long2</code>	longitude of point-1

Parameters

in	<i>real(dp) :: lat1, long1, lat2, long2</i>	latitude of point-2
in	<i>real(dp) :: lat1, long1, lat2, long2</i>	longitude of point-2
out	<i>real(dp) :: distance_out</i>	distance between two points [m]

Authors

Rohini Kumar

Date

May 2014

References `mo_constants::radiusearth_dp`, and `mo_constants::twopi_dp`.

Referenced by `celllength()`.

Here is the caller graph for this function:



15.53.2.3 l11_flow_direction()

```
subroutine, public mo_mrm_net_startup::l11_flow_direction (
    integer(i4), intent(in) iBasin )
```

Determine the flow direction of the upscaled river network at level L11.

The hydrographs generated at each cell are routed through the drainage network at level-11 towards their outlets. The drainage network at level-11 is conceptualized as a graph whose nodes are hypothetically located at the center of each grid cell connected by links that represent the river reaches. The flow direction of a link correspond to the direction towards a neighboring cell in which the net flow accumulation (outflows minus inflows) attains its maximum value. The net flow accumulation across a cell's boundary at level-11 is estimated based on flow direction and flow accumulation obtained at level-0 ([Routing Network](#)). Note: level-1 denotes the modeling level, whereas level-L11 is at least as coarse as level-1. Experience has shown that routing can be done at a coarser resolution as level-1, hence the level-11 was introduced.

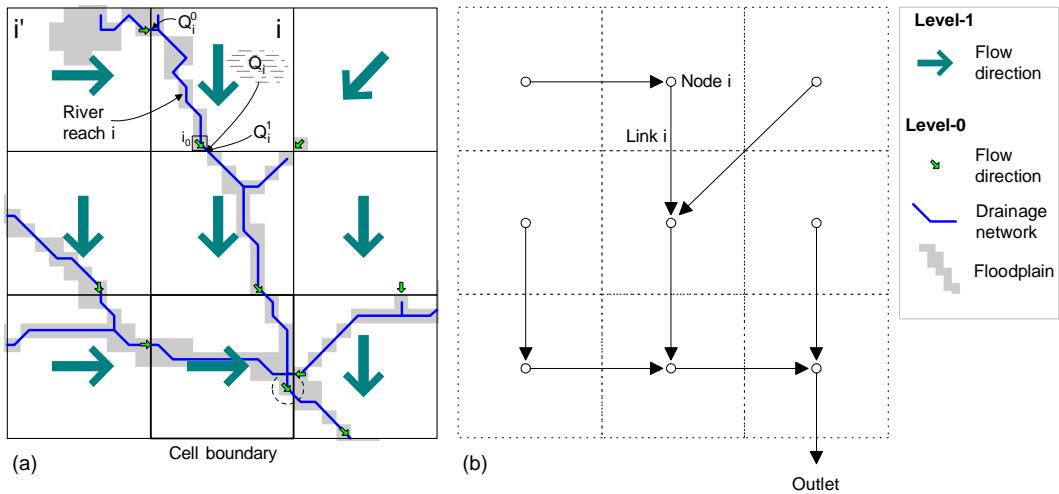


Figure 15.1: Upscaling routing network from L0 to L1 (or L11)

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 typical 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	<code>integer(i4) :: iBasin</code>	Basin Id
----	------------------------------------	----------

Authors

Luis Samaniego

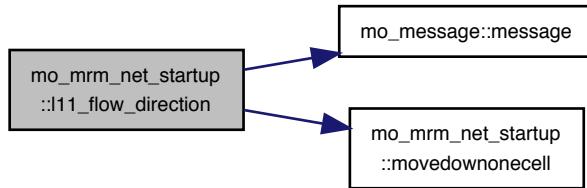
Date

Dec 2005

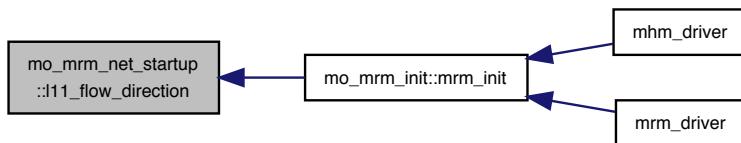
References `mo_mrm_global_variables::basin_mrm`, `mo_common_variables::l0_basin`, `mo_mrm_global_variables::l0_drasc`, `mo_mrm_global_variables::l0_facc`, `mo_mrm_global_variables::l0_fdir`, `mo_mrm_global_variables::l0_l11_remap`, `mo_mrm_global_variables::l11_colout`, `mo_mrm_global_variables::l11_fdir`, `mo_mrm_global_variables::l11_noutlets`, `mo_mrm_global_variables::l11_rowout`, `mo_common_variables::level0`, `mo_mrm_global_variables::level11`, `mo_message::message()`, `movedownonecell()`, and `mo_common_constants::nodata_i4`.

Referenced by `mo_mrm_init::mrm_init()`.

Here is the call graph for this function:



Here is the caller graph for this function:



15.53.2.4 l11_fraction_sealed_floodplain()

```

subroutine, public mo_mrm_net_startup::l11_fraction_sealed_floodplain (
    integer(i4), intent(in) LCClassImp,
    logical, intent(in) do_init )
  
```

Fraction of the flood plain with impervious cover.

Fraction of the flood plain with impervious cover (Routing Network"). This proportion is used to regionalize the Muskingum parameters. Samaniego et al. [13] 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_ncdf_restart](#)

Parameters

in	<i>integer(i4) :: LCClassImp</i>	Impervious land cover class Id, e.g. = 2 (old code)
in	<i>logical :: do_init</i>	

Authors

Luis Samaniego

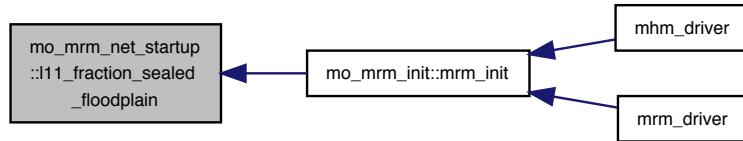
Date

Dec 2005

References mo_common_variables::l0_basin, mo_mrm_global_variables::l0_floodplain, mo_common_variables::l0_lcover, mo_mrm_global_variables::l11_afloodplain, mo_mrm_global_variables::l11_nlinkfracfpimp, mo_mrm_global_variables::l11_noutlets, mo_common_variables::level0, mo_mrm_global_variables::level11, mo_common_variables::nbasins, mo_common_variables::nlcoverscene, and mo_common_constants::nodata_dp.

Referenced by mo_mrm_init::mrm_init().

Here is the caller graph for this function:



15.53.2.5 l11_l1_mapping()

```
subroutine, public mo_mrm_net_startup::l11_l1_mapping (
    integer(i4), intent(in) iBasin )
```

TODO: add description.

TODO: add description

Parameters

in	integer(i4) :: iBasin	basin
----	-----------------------	-------

Authors

Robert Schwepppe

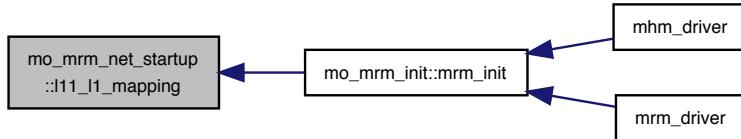
Date

Jun 2018

References `mo_kind::i4`, `mo_mrm_global_variables::l11_l1_id`, `mo_mrm_global_variables::l1_l11_id`, `mo_common_variables::level1`, `mo_mrm_global_variables::level11`, and `mo_common_constants::nodata_i4`.

Referenced by `mo_mrm_init::mrm_init()`.

Here is the caller graph for this function:



15.53.2.6 l11_link_location()

```
subroutine, public mo_mrm_net_startup::l11_link_location (
    integer(i4), intent(in) iBasin )
```

Estimate the LO (row,col) location for each routing link at level L11.

If a variable is added or removed here, then it also has to be added or removed in the subroutine `L11_config_set` in module `mo_restart` and in the subroutine `set_L11_config` in module `mo_set_netcdf_restart`

Parameters

in	<code>integer(i4) :: iBasin</code>	Basin Id
----	------------------------------------	----------

Authors

Luis Samaniego

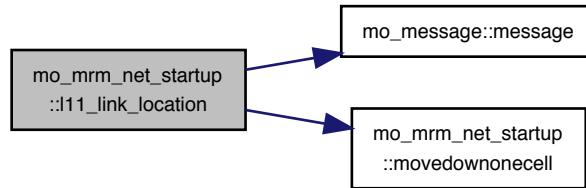
Date

Dec 2005

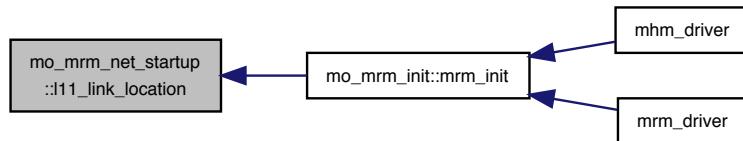
References `mo_mrm_global_variables::basin_mrm`, `mo_common_variables::l0_basin`, `mo_mrm_global_variables::l0_drasc`, `mo_mrm_global_variables::l0_fdir`, `mo_mrm_global_variables::l11_colout`, `mo_mrm_global_variables::l11_fcol`, `mo_mrm_global_variables::l11_fromn`, `mo_mrm_global_variables::l11_frow`, `mo_mrm_global_variables::l11_netperm`, `mo_mrm_global_variables::l11_noutlets`, `mo_mrm_global_variables::l11_rowout`, `mo_mrm_global_variables::l11_tcol`, `mo_mrm_global_variables::l11_trow`, `mo_common_variables::level0`, `mo_mrm_global_variables::level11`, `mo_message::message()`, `movedownonecell()`, and `mo_common_constants::nodata_i4`.

Referenced by `mo_mrm_init::mrm_init()`.

Here is the call graph for this function:



Here is the caller graph for this function:



15.53.2.7 l11_routing_order()

```
subroutine, public mo_mrm_net_startup::l11_routing_order (
    integer(i4), intent(in) iBasin )
```

Find routing order, headwater cells and sink.

Find routing order, headwater cells and sink. If a variable is added or removed here, then it also has to be added or removed in the subroutine L11_config_set in module [mo_restart](#) and in the subroutine set_L11_config in module [mo_set_ncdf_restart](#)

Parameters

in	integer(i4) :: iBasin	Basin Id
----	-----------------------	----------

Authors

Luis Samaniego

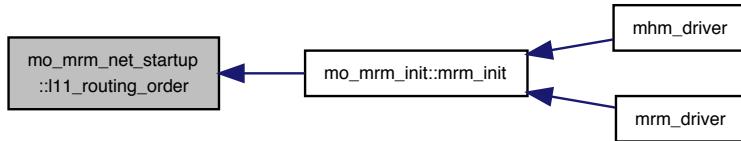
Date

Dec 2005

References `mo_mrm_global_variables::l11_fdir`, `mo_mrm_global_variables::l11_fromn`, `mo_mrm_global_variables::l11_label`, `mo_mrm_global_variables::l11_netperm`, `mo_mrm_global_variables::l11_noutlets`, `mo_mrm_global_variables::l11_order`, `mo_mrm_global_variables::l11_sink`, `mo_mrm_global_variables::l11_ton`, `mo_mrm_global_variables::level11`, and `mo_common_constants::nodata_i4`.

Referenced by `mo_mrm_init::mrm_init()`.

Here is the caller graph for this function:



15.53.2.8 l11_set_drain_outlet_gauges()

```
subroutine, public mo_mrm_net_startup::l11_set_drain_outlet_gauges (
    integer(i4), intent(in) iBasin )
```

Draining cell identification and Set gauging node.

Perform the following tasks:

- Draining cell identification (cell at L0 to draining cell outlet at L11).
- Set gauging nodes If a variable is added or removed here, then it also has to be added or removed in the subroutine `L11_config_set` in module `mo_restart` and in the subroutine `set_L11_config` in module `mo_set_netcdf_restart`

Parameters

in	<code>integer(i4) :: iBasin</code>	Basin Id
----	------------------------------------	----------

Authors

Luis Samaniego

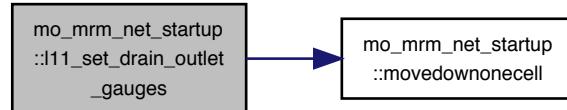
Date

Dec 2005

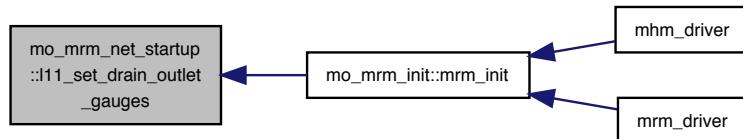
References `mo_mrm_global_variables::basin_mrm`, `mo_common_variables::l0_basin`, `mo_mrm_global_variables::l0_dracell`, `mo_mrm_global_variables::l0_drasc`, `mo_mrm_global_variables::l0_fdir`, `mo_mrm_global_variables::l0_gaugeloc`, `mo_mrm_global_variables::l0_inflowgaugeloc`, `mo_mrm_global_variables::l0_l11_remap`, `mo_common_variables::level0`, `movedownonecell()`, and `mo_common_constants::nodata_i4`.

Referenced by `mo_mrm_init::mrm_init()`.

Here is the call graph for this function:



Here is the caller graph for this function:



15.53.2.9 l11_set_network_topology()

```
subroutine, public mo_mrm_net_startup::l11_set_network_topology (
    integer(i4), intent(in) iBasin )
```

Set network topology.

Set network topology from and to node for all links at level-11 ([Routing Network](#)). If a variable is added or removed here, then it also has to be added or removed in the subroutine L11_config_set in module [mo_restart](#) and in the subroutine set_L11_config in module [mo_set_netcdf_restart](#).

Parameters

in	integer(i4) :: iBasin	Basin Id
----	-----------------------	----------

Authors

Luis Samaniego

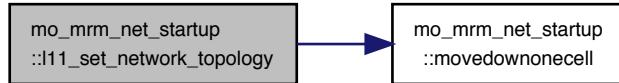
Date

Dec 2005

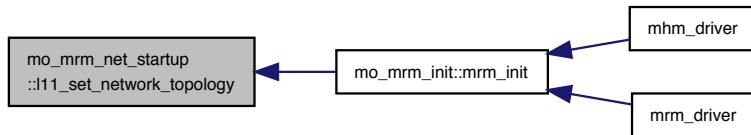
References mo_mrm_global_variables::l11_fdir, mo_mrm_global_variables::l11_fromn, mo_mrm_global_variables::l11_ton, mo_mrm_global_variables::level11, movedownonecell(), and mo_common_constants::nodata_i4.

Referenced by `mo_mrm_init::mrm_init()`.

Here is the call graph for this function:



Here is the caller graph for this function:



15.53.2.10 l11_stream_features()

```
subroutine, public mo_mrm_net_startup::l11_stream_features (
    integer(i4), intent(in) iBasin )
```

Stream features (stream network and floodplain)

Stream features (stream network and floodplain) If a variable is added or removed here, then it also has to be added or removed in the subroutine `L11_config_set` in module `mo_restart` and in the subroutine `set_L11_config` in module `mo_set_ncdf_restart`

Parameters

in	<code>integer(i4) :: iBasin</code>	Basin Id
----	------------------------------------	----------

Authors

Luis Samaniego

Date

Dec 2005

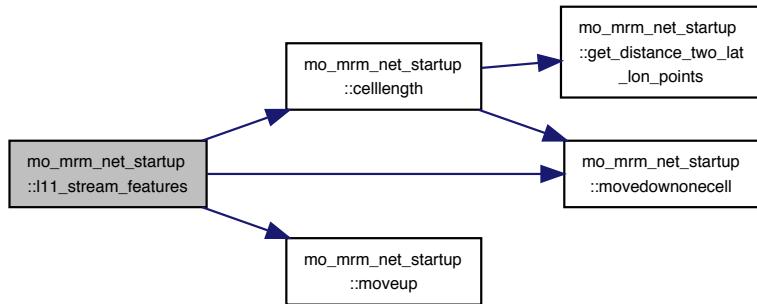
`stack(nNodes, 2)`

References `celllength()`, `mo_common_variables::iflag_cordinate_sys`, `mo_common_variables::l0_basin`, `mo_common_variables::l0_elev`, `mo_mrm_global_variables::l0_fdir`, `mo_mrm_global_variables::l0_floodplain`, `mo`

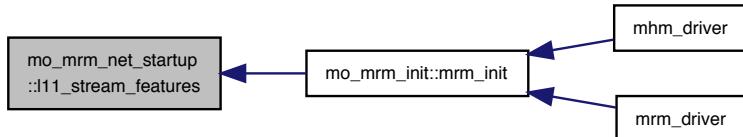
_mrm_global_variables::l0_streamnet, mo_mrm_global_variables::l11_afloodplain, mo_mrm_global_variables::l11_fcol, mo_mrm_global_variables::l11_frow, mo_mrm_global_variables::l11_length, mo_mrm_global_variables::l11_netperm, mo_mrm_global_variables::l11_noutlets, mo_mrm_global_variables::l11_slope, mo_mrm_global_variables::l11_tcol, mo_mrm_global_variables::l11_trow, mo_common_variables::level0, mo_mrm_global_variables::level11, movedownonecell(), moveup(), mo_common_constants::nodata_dp, mo_common_constants::nodata_i4, and mo_common_variables::processmatrix.

Referenced by mo_mrm_init::mrm_init().

Here is the call graph for this function:



Here is the caller graph for this function:



15.53.2.11 movedownonecell()

```

subroutine mo_mrm_net_startup::movedownonecell (
    integer(i4), intent(in) fDir,
    integer(i4), intent(inout) iRow,
    integer(i4), intent(inout) jCol )

```

TODO: add description.

TODO: add description

Parameters

in	integer(i4) :: fDir	
in, out	integer(i4) :: iRow, jCol	

Parameters

in, out	integer(i4) :: iRow, jCol	
---------	---------------------------	--

Authors

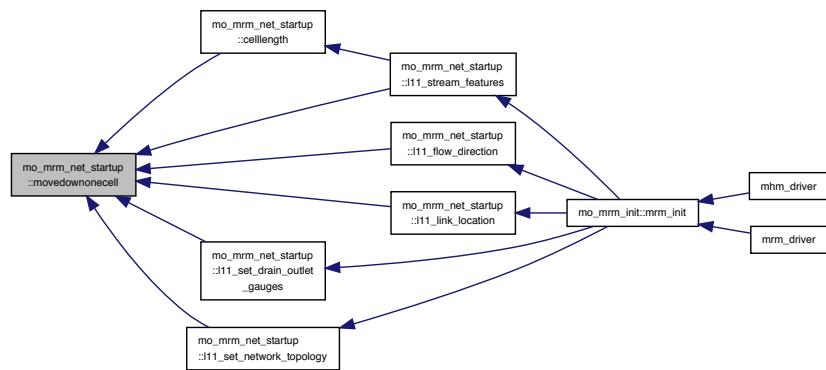
Robert Schweppel

Date

Jun 2018

Referenced by celllength(), 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:



15.53.2.12 moveup()

```

subroutine mo_mrm_net_startup::moveup (
    real(dp), dimension(:, :, ), intent(in), allocatable elev0,
    integer(i4), dimension(:, :, ), intent(in), allocatable fDir0,
    integer(i4), intent(in) fi,
    integer(i4), intent(in) fj,
    integer(i4), dimension(:, :, ), intent(inout) ss,
    integer(i4), intent(inout) nn )

```

TODO: add description.

TODO: add description

Parameters

in	real(dp), dimension(:, :,) :: elev0	
in	integer(i4), dimension(:, :,) :: fDir0	
in	integer(i4) :: fi, fj	co-ordinate of the stream bed
in	integer(i4) :: fi, fj	co-ordinate of the stream bed
in, out	integer(i4), dimension(:, :,) :: ss	
in, out	integer(i4) :: nn	

Authors

Robert Schweppe

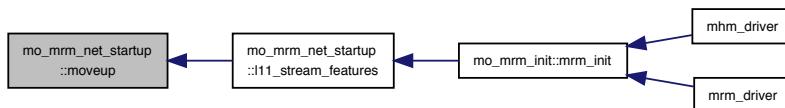
Date

Jun 2018

References mo_mrm_constants::deltah.

Referenced by l11_stream_features().

Here is the caller graph for this function:



15.54 mo_mrm_objective_function_runoff Module Reference

Objective Functions for Optimization of mHM/mRM against runoff.

Functions/Subroutines

- real(dp) function, public [single_objective_runoff](#) (parameterset, eval, arg1, arg2, arg3)
Wrapper for objective functions optimizing against 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` (gaugeld, runoff, runoff_agg, runoff_obs, runoff_obs_mask)
extracts runoff data from global variables

15.54.1 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) (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

15.54.2 Function/Subroutine Documentation

15.54.2.1 `extract_runoff()`

```
subroutine, public mo_mrm_objective_function_runoff::extract_runoff (
    integer(i4), intent(in) gaugeId,
    real(dp), dimension(:, :), intent(in) runoff,
    real(dp), dimension(:, ), intent(out), allocatable runoff_agg,
    real(dp), dimension(:, ), intent(out), allocatable runoff_obs,
    logical, dimension(:, ), intent(out), allocatable runoff_obs_mask )
```

extracts runoff data from global variables

extracts simulated and measured runoff from global variables, such that they overlay exactly. For measured runoff, only the runoff during the evaluation period are cut, not succeeding nodata values. For simulated runoff, warming days as well as succeeding nodata values are neglected and the simulated runoff is aggregated to the resolution of the observed runoff. see use in this module above

Parameters

in	<i>integer(i4) :: gaugeld</i>	current gauge Id
in	<i>real(dp), dimension(:, :) :: runoff</i>	simulated runoff
out	<i>real(dp), dimension(:, :) :: runoff_agg</i>	aggregated simulated
out	<i>real(dp), dimension(:, :) :: runoff_obs</i>	extracted measured
out	<i>logical, dimension(:, :) :: runoff_obs_mask</i>	mask of no data values

Authors

Stephan Thober

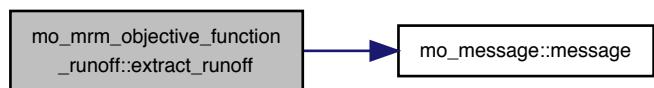
Date

Jan 2015

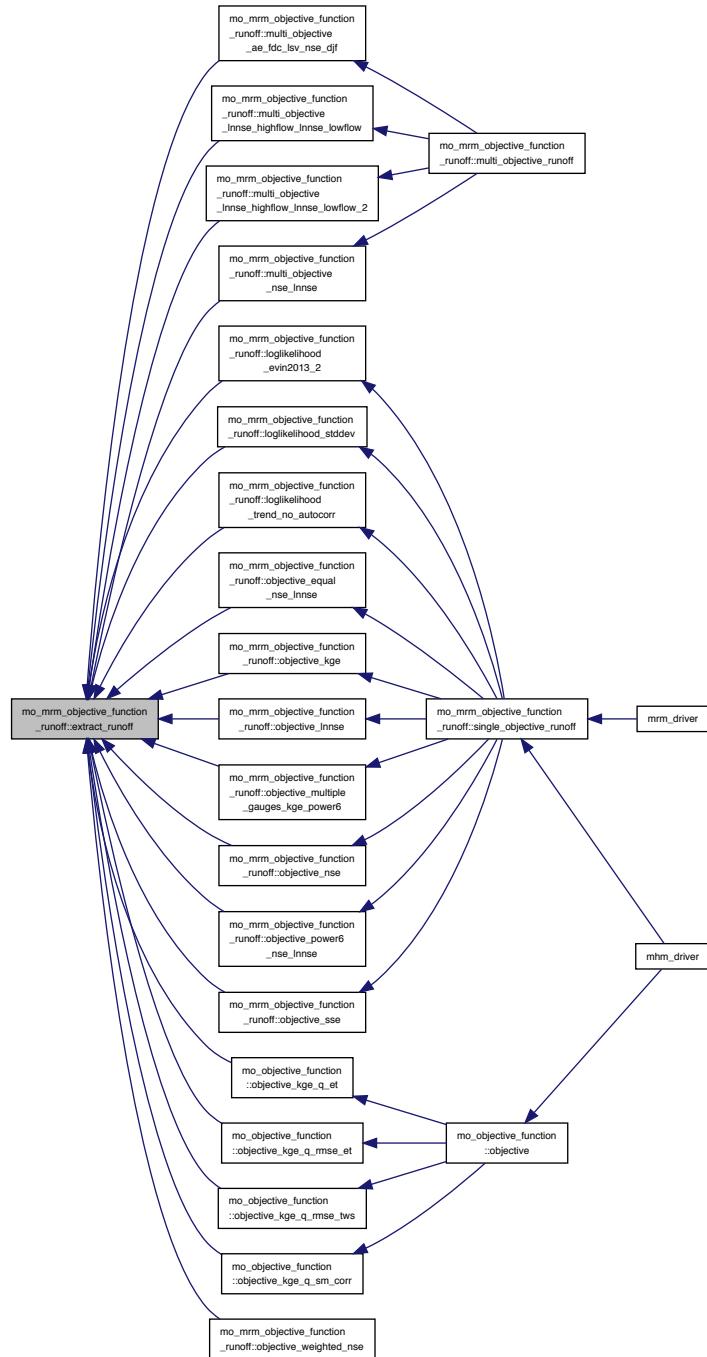
References mo_common_mhm_mrm_variables::evalper, mo_mrm_global_variables::gauge, mo_message::message(), mo_mrm_global_variables::nmeasperday, mo_common_mhm_mrm_variables::ntstepday, and mo_common_mhm_mrm_variables::warmingdays.

Referenced by loglikelihood_evin2013_2(), loglikelihood_stddev(), loglikelihood_trend_no_autocorr(), multi_objective_ae_fdc_lsv_nse_djf(), multi_objective_lnnse_highflow_lnnse_lowflow(), multi_objective_lnnse_highflow_lnnse_lowflow_2(), multi_objective_nse_lnnse(), objective_equal_nse_lnnse(), objective_kge(), mo_objective_function::objective_kge_q_et(), mo_objective_function::objective_kge_q_rmse_et(), mo_objective_function::objective_kge_q_rmse_tws(), mo_objective_function::objective_kge_q_sm_corr(), objective_lnnse(), objective_multiple_gauges_kge_power6(), objective_nse(), objective_power6_nse_lnnse(), objective_sse(), and objective_weighted_nse().

Here is the call graph for this function:



Here is the caller graph for this function:



15.54.2.2 loglikelihood_evin2013_2()

```
real(dp) function mo_mrm_objective_function_runoff::loglikelihood_evin2013_2 (
    real(dp), dimension(:), intent(in) parameterset,
    procedure(eval_interface), intent(in), pointer eval,
```

```
logical, intent(in), optional regularize )
```

Logarithmised likelihood with linear error model and lag(1)-autocorrelation of the relative errors.

This loglikelihood uses a linear error model and a lag(1)-autocorrelation on the relative errors. This is approach 2 of the paper Evin et al. (WRR, 2013). This is opti_function = 8. mHM then adds two extra (local) parameters for the error model in mhm_driver, which get optimised together with the other, global parameters. ADDITIONAL INFORMATION Evin et al., WRR 49, 4518-4524, 2013

Parameters

in	<i>real(dp), dimension(:) :: parameterset</i>	
in	<i>procedure(eval_interface) :: eval</i>	
in	<i>logical, optional :: regularize</i>	

Returns

real(dp) :: loglikelihood_evin2013_2 — logarithmic likelihood using given stddev but remove optimal trend and lag(1)-autocorrelation in errors (absolute between running model with parameterset and observation)

Authors

Juliane Mai and Matthias Cuntz

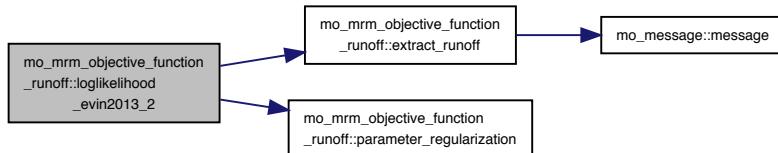
Date

Mar 2014

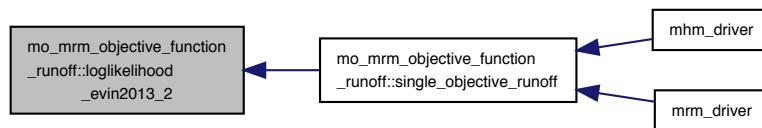
References `extract_runoff()`, `mo_common_variables::global_parameters`, `parameter_regularization()`, and `mo_constants::pi_dp`.

Referenced by `single_objective_runoff()`.

Here is the call graph for this function:



Here is the caller graph for this function:



15.54.2.3 loglikelihood_stddev()

```
real(dp) function mo_mrm_objective_function_runoff::loglikelihood_stddev (
    real(dp), dimension(:), intent(in) parameterset,
    procedure(eval_interface), intent(in), pointer eval,
    real(dp), intent(in) stddev,
    real(dp), intent(out), optional stddev_new,
    real(dp), intent(out), optional likeli_new )
```

Logarithmic likelihood function with removed linear trend and Lag(1)-autocorrelation.

The logarithmic 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	<i>real(dp), dimension(:) :: parameterset</i>	
in	<i>procedure(eval_interface) :: eval</i>	
in	<i>real(dp) :: stddev</i>	standard deviation of data
out	<i>real(dp), optional :: stddev_new</i>	standard deviation of errors with removed trend and correlation between model run using parameter set and observation
out	<i>real(dp), optional :: likeli_new</i>	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 parameter set and observation)

Authors

Juliane Mai

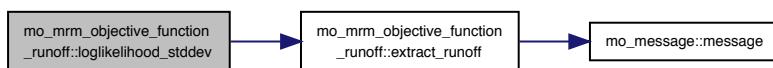
Date

Dec 2012

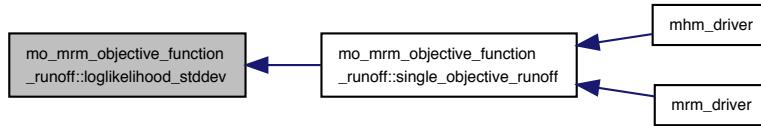
References `extract_runoff()`.

Referenced by `single_objective_runoff()`.

Here is the call graph for this function:



Here is the caller graph for this function:



15.54.2.4 loglikelihood_trend_no_autocorr()

```
real(dp) function mo_mrm_objective_function_runoff::loglikelihood_trend_no_autocorr (
    real(dp), dimension(:), intent(in) parameterset,
    procedure(eval_interface), intent(in), pointer eval,
    real(dp), intent(in) stddev_old,
    real(dp), intent(out), optional stddev_new,
    real(dp), intent(out), optional likeli_new )
```

Logarithmic likelihood function with linear trend removed.

The logarithmic 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	<i>real(dp)</i> , <i>dimension(:)</i> :: <i>parameterset</i>	
in	<i>procedure(eval_interface)</i> :: <i>eval</i>	
in	<i>real(dp)</i> :: <i>stddev_old</i>	standard deviation of data
out	<i>real(dp)</i> , <i>optional</i> :: <i>stddev_new</i>	standard deviation of errors with removed trend between model run using parameter set and observation
out	<i>real(dp)</i> , <i>optional</i> :: <i>likeli_new</i>	logarithmic likelihood determined with <i>stddev_new</i> instead of <i>stddev</i>

Returns

real(dp) :: *loglikelihood_trend_no_autocorr* — logarithmic likelihood using given *stddev* but remove optimal trend in errors (absolute between running model with *parameterset* and observation)

Authors

Juliane Mai and Matthias Cuntz

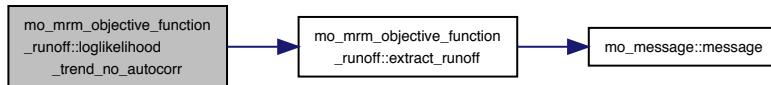
Date

Mar 2014

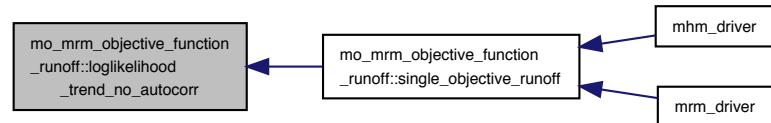
References *extract_runoff()*.

Referenced by *single_objective_runoff()*.

Here is the call graph for this function:



Here is the caller graph for this function:



15.54.2.5 multi_objective_ae_fdc_lsv_nse_djf()

```

real(dp) function, dimension(2) mo_mrm_objective_function_runoff::multi_objective_ae_fdc_lsv_nse_djf (
    real(dp), dimension(:), intent(in) parameteriset,
    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	<code>real(dp), dimension(:) :: parameteriset</code>	
in	<code>procedure(eval_interface) :: eval</code>	

Returns

`real(dp), dimension(2) :: multi_objective_ae_fdc_lsv_nse_djf` — objective function value (which will be e.g. minimized by an optimization routine like PA-DDS)

Authors

Juliane Mai

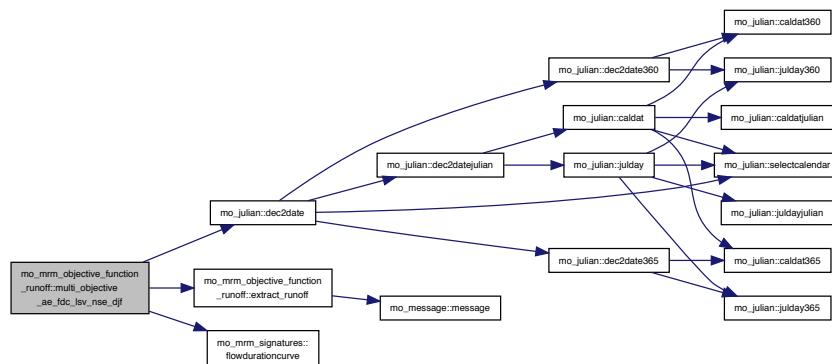
Date

Feb 2016

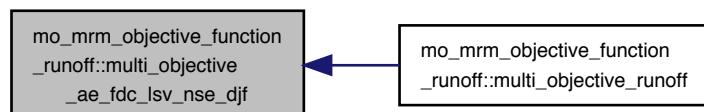
References `mo Julian::dec2date()`, `mo_common_mhm_mrm_variables::evalper`, `extract_runoff()`, `mo_mrm_signatures::flowdurationcurve()`, `mo_mrm_global_variables::gauge`, and `mo_mrm_global_variables::nmeasperday`.

Referenced by `multi_objective_runoff()`.

Here is the call graph for this function:



Here is the caller graph for this function:



15.54.2.6 multi_objective_lnnse_highflow_lnnse_lowflow()

```
real(dp) function, dimension(2) mo_mrm_objective_function_runoff::multi_objective_lnnse_lowflow
highflow_lnnse_lowflow (
    real(dp), dimension(:), intent(in) parameterset,
    procedure(eval_interface), intent(in), pointer eval )
```

Multi-objective function with NSE and InNSE.

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

$$t_{high} \text{ if } Q_{obs}(i) > \text{percentile}(Q_{obs}, 95.)$$

This timepoint identification is only performed for the observed data. The first objective is the logarithmic Nash-Sutcliffe model efficiency coefficient InNSE_{high} of discharge values at high-flow timepoints

$$\text{InNSE}_{high} = 1 - \frac{\sum_{i=1}^{N_{high}} (\ln Q_{obs}(i) - \ln Q_{model}(i))^2}{\sum_{i=1}^N (\ln Q_{obs}(i) - \ln \bar{Q}_{obs})^2}$$

. The second objective is the logarithmic Nash-Sutcliffe model efficiency coefficient InNSE_{low} of discharge values at low-flow timepoints

$$\text{InNSE}_{low} = 1 - \frac{\sum_{i=1}^{N_{low}} (\ln Q_{obs}(i) - \ln Q_{model}(i))^2}{\sum_{i=1}^N (\ln Q_{obs}(i) - \ln \bar{Q}_{obs})^2}$$

. 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	<code>real(dp), dimension(:) :: parameterset</code>	
in	<code>procedure(eval_interface) :: eval</code>	

Returns

`real(dp), dimension(2) :: multi_objective_Innse_highflow_Innse_lowflow` — objective function value (which will be e.g. minimized by an optimization routine like PA-DDS)

Authors

Juliane Mai

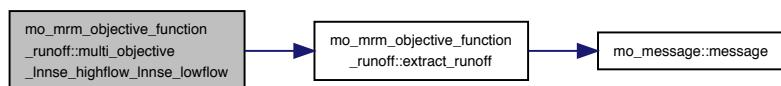
Date

Oct 2015

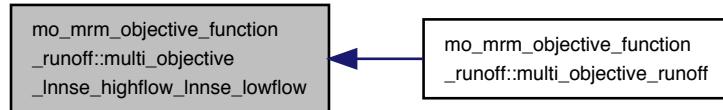
References `extract_runoff()`.

Referenced by `multi_objective_runoff()`.

Here is the call graph for this function:



Here is the caller graph for this function:



15.54.2.7 multi_objective_lnnse_highflow_lnnse_lowflow_2()

```
real(dp) function, dimension(2) mo_mrm_objective_function_runoff::multi_objective_lnnse_lowflow_2
highflow_lnnse_lowflow_2 (
    real(dp), dimension(:), intent(in) parameterset,
    procedure(eval_interface), intent(in), pointer eval )
```

Multi-objective function with NSE and lnNSE.

The objective function only depends on a parameter vector. The model will be called with that parameter vector and the model output is subsequently compared to observed data. A timepoint t of the observed data is marked as a lowflow timepoint t_{low} if

$$Q_{obs}(t) < \min(Q_{obs}) + 0.05 * (\max(Q_{obs}) - \min(Q_{obs}))$$

and all other timepoints are marked as a highflow timepoints t_{high} . This timepoint identification is only performed for the observed data. The first objective is the logarithmic Nash-Sutcliffe model efficiency coefficient $lnNSE_{high}$ of discharge values at high-flow timepoints

$$lnNSE_{high} = 1 - \frac{\sum_{i=1}^{N_{high}} (\ln Q_{obs}(i) - \ln Q_{model}(i))^2}{\sum_{i=1}^N (\ln Q_{obs}(i) - \ln \bar{Q}_{obs})^2}$$

. The second objective is the logarithmic Nash-Sutcliffe model efficiency coefficient $lnNSE_{low}$ of discharge values at low-flow timepoints

$$lnNSE_{low} = 1 - \frac{\sum_{i=1}^{N_{low}} (\ln Q_{obs}(i) - \ln Q_{model}(i))^2}{\sum_{i=1}^N (\ln Q_{obs}(i) - \ln \bar{Q}_{obs})^2}$$

. Both objectives are returned. The observed data Q_{obs} are global in this module. To calibrate this objective you need a multi-objective optimizer like PA-DDS.

Parameters

in	real(dp), dimension(:) :: parameterset	
in	procedure(eval_interface) :: eval	

Returns

real(dp), dimension(2) :: multi_objective_lnnse_highflow_lnnse_lowflow_2 — objective function value (which will be e.g. minimized by an optimization routine like PA-DDS)

Authors

Juliane Mai

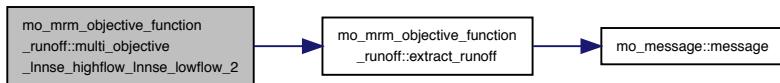
Date

Oct 2015

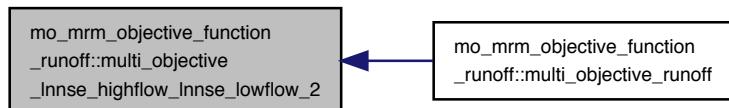
References extract_runoff().

Referenced by multi_objective_runoff().

Here is the call graph for this function:



Here is the caller graph for this function:



15.54.2.8 multi_objective_nse_lnnse()

```

real(dp) function, dimension(2) mo_mrm_objective_function_runoff::multi_objective_nse_lnnse (
    real(dp), dimension(:), intent(in) parameterset,
    procedure(eval_interface), intent(in), pointer eval )

```

Multi-objective function with NSE and lnNSE.

The objective function only depends on a parameter vector. The model will be called with that parameter vector and the model output is subsequently compared to observed data. Therefore, the Nash-Sutcliffe model efficiency coefficient *NSE*

$$NSE = 1 - \frac{\sum_{i=1}^N (Q_{obs}(i) - Q_{model}(i))^2}{\sum_{i=1}^N (Q_{obs}(i) - \bar{Q}_{obs})^2}$$

and the logarithmic Nash-Sutcliffe model efficiency coefficient *lnNSE*

$$lnNSE = 1 - \frac{\sum_{i=1}^N (\ln Q_{obs}(i) - \ln Q_{model}(i))^2}{\sum_{i=1}^N (\ln Q_{obs}(i) - \ln \bar{Q}_{obs})^2}$$

are calculated and both returned. The observed data Q_{obs} are global in this module. To calibrate this objective you need a multi-objective optimizer like PA-DDS.

Parameters

in	real(dp), dimension(:) :: parameterset	
in	procedure(eval_interface) :: eval	

Returns

real(dp), dimension(2) :: multi_objective_nse_lnnse — objective function value (which will be e.g. minimized by an optimization routine like PA-DDS)

Authors

Juliane Mai

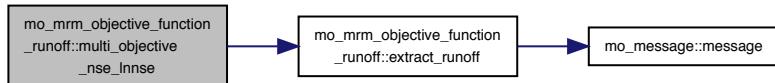
Date

Oct 2015

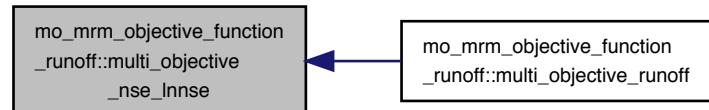
References extract_runoff().

Referenced by multi_objective_runoff().

Here is the call graph for this function:



Here is the caller graph for this function:



15.54.2.9 multi_objective_runoff()

```

subroutine, public mo_mrm_objective_function_runoff::multi_objective_runoff (
    real(dp), dimension(:, ), intent(in) parameterset,
    procedure(eval_interface), intent(in), pointer eval,
    real(dp), dimension(:, ), intent(out), allocatable multi_objectives )
  
```

Wrapper for multi-objective functions where at least one is regarding runoff.

The functions selects the objective function case defined in a namelist, i.e. the global variable *opti_function*. It return the multiple objective function values for a specific parameter set.

Parameters

in	<i>REAL(dp), DIMENSION(:) :: parameterset</i>
in	<i>procedure(eval_interface) :: eval</i>
out	<i>REAL(dp), DIMENSION(:) :: multi_objectives</i>

Authors

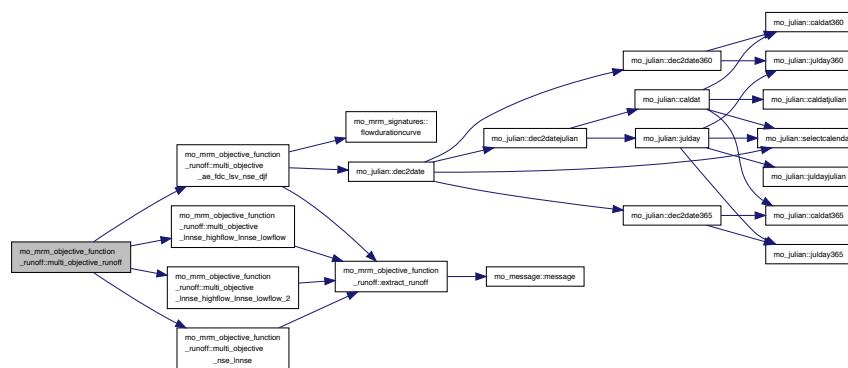
Juliane Mai

Date

Oct 2015

References `multi_objective_ae_fdc_lsv_nse_djf()`, `multi_objective_lnnse_highflow_lnnse_lowflow()`, `multi_objective_lnnse_highflow_lnnse_lowflow_2()`, `multi_objective_nse_lnnse()`, and `mo_common_mhm_mrm::variables::opti_function`.

Here is the call graph for this function:

15.54.2.10 `objective_equal_nse_lnnse()`

```
real(dp) function mo_mrm_objective_function_runoff::objective_equal_nse_lnnse (
    real(dp), dimension(:), intent(in) parameterset,
    procedure(eval_interface), intent(in), pointer eval )
```

Objective function equally weighting NSE and InNSE.

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*

$$lnNSE = 1 - \frac{\sum_{i=1}^N (\ln Q_{obs}(i) - \ln Q_{model}(i))^2}{\sum_{i=1}^N (\ln Q_{obs}(i) - \bar{\ln Q}_{obs})^2}$$

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	<code>real(dp), dimension(:) :: parameterset</code>	
in	<code>procedure(eval_interface) :: eval</code>	

Returns

`real(dp) :: objective_equal_nse_Innse` — objective function value (which will be e.g. minimized by an optimization routine like DDS)

Authors

Juliane Mai

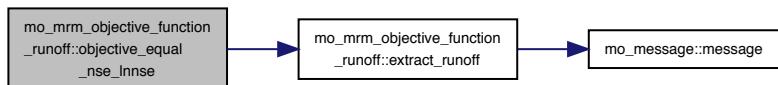
Date

May 2013

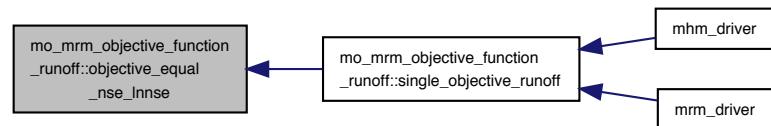
References `extract_runoff()`.

Referenced by `single_objective_runoff()`.

Here is the call graph for this function:



Here is the caller graph for this function:

15.54.2.11 `objective_kge()`

```
real(dp) function mo_mrm_objective_function_runoff::objective_kge (
    real(dp), dimension(:), intent(in) parameterset,
    procedure(eval_interface), intent(in), pointer eval )
```

Objective function of KGE.

The objective function only depends on a parameter vector. The model will be called with that parameter vector and the model output is subsequently compared to observed data. Therefore, the Kling-Gupta model efficiency coefficient KGE

$$KGE = 1.0 - \sqrt{((1-r)^2 + (1-\alpha)^2 + (1-\beta)^2)}$$

where r = Pearson product-moment correlation coefficient α = ratio of simulated mean to observed mean β = ratio of simulated 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	<code>real(dp), dimension(:) :: parameterset</code>	
in	<code>procedure(eval_interface) :: eval</code>	

Returns

`real(dp) :: objective_kge` — objective function value (which will be e.g. minimized by an optimization routine like DDS)

Authors

Rohini Kumar

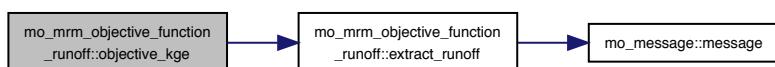
Date

August 2014

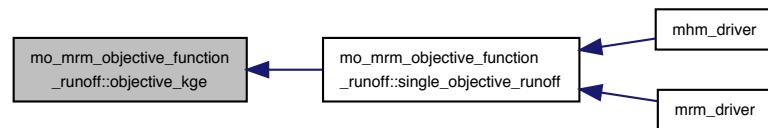
References `extract_runoff()`.

Referenced by `single_objective_runoff()`.

Here is the call graph for this function:



Here is the caller graph for this function:



15.54.2.12 `objective_lnnse()`

```
real(dp) function mo_mrm_objective_function_runoff::objective_lnnse (
    real(dp), dimension(:), intent(in) parameterset,
    procedure(eval_interface), intent(in), pointer eval )
```

Objective function of logarithmic NSE.

The objective function only depends on a parameter vector. The model will be called with that parameter vector and the model output is subsequently compared to observed data. Therefore, the logarithmic Nash-Sutcliffe model efficiency coefficient $lnNSE$

$$lnNSE = 1 - \frac{\sum_{i=1}^N (\ln Q_{obs}(i) - \ln Q_{model}(i))^2}{\sum_{i=1}^N (\ln Q_{obs}(i) - \ln \bar{Q}_{obs})^2}$$

is calculated.

$$obj_value = lnNSE$$

The observed data Q_{obs} are global in this module.

Parameters

in	<code>real(dp), dimension(:) :: parameterset</code>	
in	<code>procedure(eval_interface) :: eval</code>	

Returns

`real(dp) :: objective_lnnse` — objective function value (which will be e.g. minimized by an optimization routine like DDS)

Authors

Juliane Mai

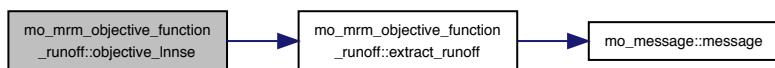
Date

May 2013

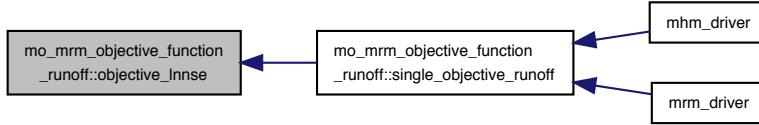
References `extract_runoff()`.

Referenced by `single_objective_runoff()`.

Here is the call graph for this function:



Here is the caller graph for this function:



15.54.2.13 objective_multiple_gauges_kge_power6()

```
real(dp) function mo_mrm_objective_function_runoff::objective_multiple_gauges_kge_power6 (
    real(dp), dimension(:), intent(in) parameterset,
    procedure(eval_interface), intent(in), pointer eval )
```

combined objective function based on KGE raised to the power 6

The objective function only depends on a parameter vector. The model will be called with that parameter vector and the model output is subsequently compared to observed data. Therefore, the Kling-Gupta model efficiency coefficient *KGE* for a given gauging station

$$KGE = 1.0 - \sqrt{((1 - r)^2 + (1 - \alpha)^2 + (1 - \beta)^2)}$$

where r = Pearson product-moment correlation coefficient α = ratio of simulated mean to observed mean β = ratio of simulated standard deviation to observed standard deviation is calculated and the objective function for a given gauging station (i) is

$$\phi_i = 1.0 - KGE_i$$

ϕ_i is the objective since we always apply minimization methods. The minimal value of ϕ_i is 0 for the optimal KGE of 1.0. Finally, the overall *OF* is estimated based on the power-6 norm to combine the ϕ_i from all gauging stations (N).

$$OF = \sqrt[6]{\sum ((1.0 - KGE_i)/N)^6}$$

. The observed data Q_{obs} are global in this module.

Parameters

in	real(dp), dimension(:) :: parameterset	
in	procedure(eval_interface) :: eval	

Returns

real(dp) :: objective_multiple_gauges_kge_power6 — objective function value (which will be e.g. minimized by an optimization routine like DDS)

Authors

Rohini Kumar

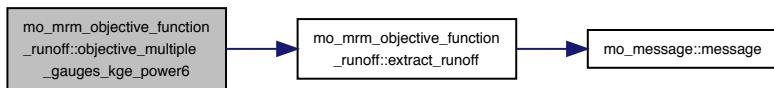
Date

March 2015

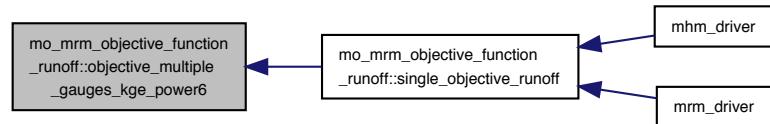
References `extract_runoff()`.

Referenced by `single_objective_runoff()`.

Here is the call graph for this function:



Here is the caller graph for this function:



15.54.2.14 `objective_nse()`

```
real(dp) function mo_mrm_objective_function_runoff::objective_nse (
    real(dp), dimension(:), intent(in) parameterset,
    procedure(eval_interface), intent(in), pointer eval )
```

Objective function of NSE.

The objective function only depends on a parameter vector. The model will be called with that parameter vector and the model output is subsequently compared to observed data. Therefore, the Nash-Sutcliffe model efficiency coefficient *NSE*

$$NSE = 1 - \frac{\sum_{i=1}^N (Q_{obs}(i) - Q_{model}(i))^2}{\sum_{i=1}^N (Q_{obs}(i) - \bar{Q}_{obs})^2}$$

is calculated and the objective function is

$$obj_value = 1 - NSE$$

The observed data Q_{obs} are global in this module.

Parameters

in	<code>real(dp), dimension(:) :: parameterset</code>	
in	<code>procedure(eval_interface) :: eval</code>	

Returns

real(dp) :: objective_nse — objective function value (which will be e.g. minimized by an optimization routine like DDS)

Authors

Juliane Mai

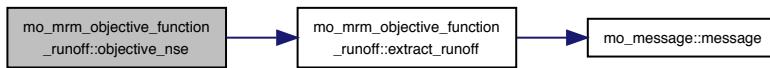
Date

May 2013

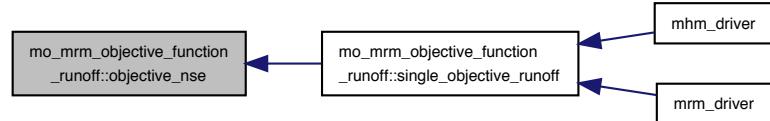
References extract_runoff().

Referenced by single_objective_runoff().

Here is the call graph for this function:



Here is the caller graph for this function:



15.54.2.15 objective_power6_nse_lnnse()

```
real(dp) function mo_mrm_objective_function_runoff::objective_power6_nse_lnnse (
    real(dp), dimension(:), intent(in) parameterset,
    procedure(eval_interface), intent(in), pointer eval )
```

Objective function of combined NSE and InNSE 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 *InNSE*

$$InNSE = 1 - \frac{\sum_{i=1}^N (\ln Q_{obs}(i) - \ln Q_{model}(i))^2}{\sum_{i=1}^N (\ln Q_{obs}(i) - \bar{\ln Q}_{obs})^2}$$

are calculated and added up equally weighted:

$$obj_value = \sqrt[6]{(1 - NSE)^6 + (1 - \ln NSE)^6}$$

The observed data Q_{obs} are global in this module.

Parameters

in	<code>real(dp), dimension(:) :: parameterset</code>	
in	<code>procedure(eval_interface) :: eval</code>	

Returns

`real(dp) :: objective_power6_nse_lnse` — objective function value (which will be e.g. minimized by an optimization routine like DDS)

Authors

Juliane Mai and Matthias Cuntz

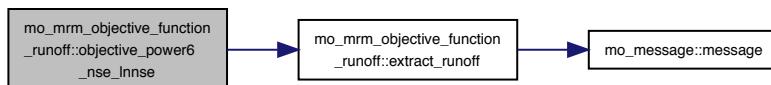
Date

March 2014

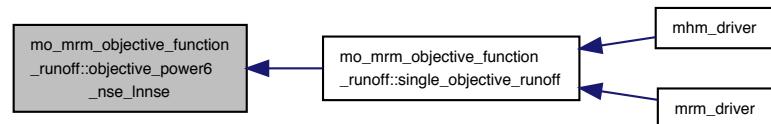
References `extract_runoff()`.

Referenced by `single_objective_runoff()`.

Here is the call graph for this function:



Here is the caller graph for this function:



15.54.2.16 objective_sse()

```
real(dp) function mo_mrm_objective_function_runoff::objective_sse (
    real(dp), dimension(:), intent(in) parameterset,
    procedure(eval_interface), intent(in), pointer eval )
```

Objective function of SSE.

The objective function only depends on a parameter vector. The model will be called with that parameter vector and the model output is subsequently compared to observed data. Therefore, the sum squared errors

$$SSE = \sum_{i=1}^N (Q_{obs}(i) - Q_{model}(i))^2$$

is calculated and the objective function is

$$obj_value = SSE$$

The observed data Q_{obs} are global in this module.

Parameters

in	real(dp), dimension(:) :: parameterset	
in	procedure(eval_interface) :: eval	

Returns

real(dp) :: objective_sse — objective function value (which will be e.g. minimized by an optimization routine like DDS)

Authors

Juliane Mai and Matthias Cuntz

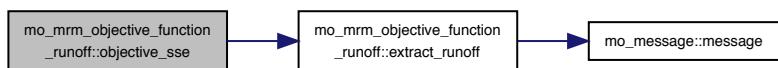
Date

March 2014

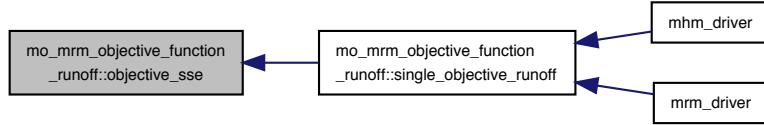
References extract_runoff().

Referenced by single_objective_runoff().

Here is the call graph for this function:



Here is the caller graph for this function:



15.54.2.17 objective_weighted_nse()

```
real(dp) function mo_mrm_objective_function_runoff::objective_weighted_nse (
    real(dp), dimension(:), intent(in) parameterset,
    procedure(eval_interface), intent(in), pointer eval )
```

Objective function of weighted NSE.

The objective function only depends on a parameter vector. The model will be called with that parameter vector and the model output is subsequently compared to observed data. Therefore, the weighted Nash-Sutcliffe model efficiency coefficient NSE

$$wNSE = 1 - \frac{\sum_{i=1}^N Q_{obs}(i) * (Q_{obs}(i) - Q_{model}(i))^2}{\sum_{i=1}^N Q_{obs}(i) * (Q_{obs}(i) - \bar{Q}_{obs})^2}$$

is calculated and the objective function is

$$obj_value = 1 - wNSE$$

The observed data Q_{obs} are global in this module.

Parameters

in	real(dp), dimension(:) :: parameterset	
in	procedure(eval_interface) :: eval	

Returns

real(dp) :: objective_weighted_nse — objective function value (which will be e.g. minimized by an optimization routine like DDS)

Authors

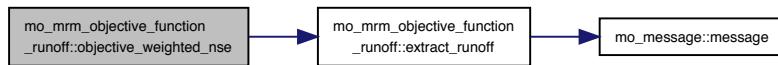
Stephan Thober, Bjoern Guse

Date

May 2018

References extract_runoff().

Here is the call graph for this function:



15.54.2.18 parameter_regularization()

```

real(dp) function mo_mrm_objective_function_runoff::parameter_regularization (
    real(dp), dimension(:), intent(in) paraset,
    real(dp), dimension(size(paraset)), intent(in) prior,
    real(dp), dimension(size(paraset), 2), intent(in) bounds,
    logical, dimension(size(paraset)), intent(in) mask )
  
```

TODO: add description.

TODO: add description

Parameters

in	<i>real(dp), dimension(:) :: paraset</i>	
in	<i>real(dp), dimension(size(paraset)) :: prior</i>	
in	<i>real(dp), dimension(size(paraset), 2) :: bounds</i>	(min, max)
in	<i>logical, dimension(size(paraset)) :: mask</i>	

Authors

Robert Scheweppe

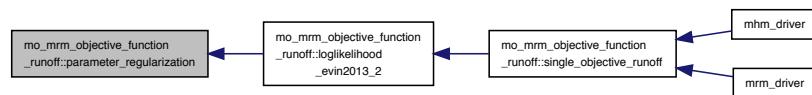
Date

Jun 2018

References mo_constants::pi_dp.

Referenced by loglikelihood_evin2013_2().

Here is the caller graph for this function:



15.54.2.19 single_objective_runoff()

```
real(dp) function, public mo_mrm_objective_function_runoff::single_objective_runoff (
    real(dp), dimension(:), intent(in) parameterset,
    procedure(eval_interface), intent(in), pointer eval,
    real(dp), intent(in), optional arg1,
    real(dp), intent(out), optional arg2,
    real(dp), intent(out), optional arg3 )
```

Wrapper for objective functions optimizing against runoff.

The function selects the objective function case defined in a namelist, i.e. the global variable *opti_function*. It returns the objective function value for a specific parameter set.

Parameters

in	<i>REAL(dp), DIMENSION(:) :: parameterset</i>	
in	<i>procedure(eval_interface) :: eval</i>	
in	<i>real(dp), optional :: arg1</i>	
out	<i>real(dp), optional :: arg2</i>	
out	<i>real(dp), optional :: arg3</i>	

Returns

real(dp) :: objective — objective function value (which will be e.g. minimized by an optimization routine like DDS)

Authors

Juliane Mai

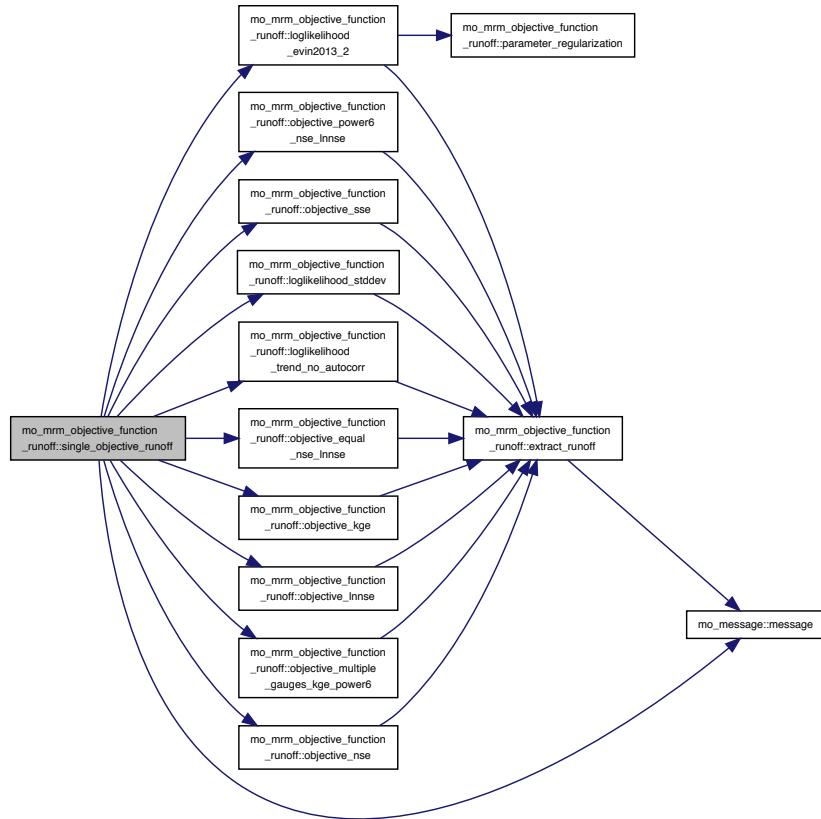
Date

Dec 2012

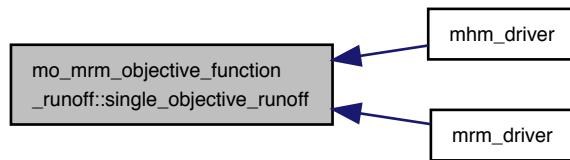
References loglikelihood_evin2013_2(), loglikelihood_stddev(), loglikelihood_trend_no_autocorr(), mo_message::message(), objective_equal_nse_lnnse(), objective_kge(), objective_lnnse(), objective_multiple_gauges_kge_power6(), objective_nse(), objective_power6_nse_lnnse(), objective_sse(), mo_common_mhm_mrm_variables::opti_function, and mo_common_mhm_mrm_variables::opti_method.

Referenced by mhm_driver(), and mrm_driver().

Here is the call graph for this function:



Here is the caller graph for this function:



15.55 mo_mrm_read_config Module Reference

read mRM config

Functions/Subroutines

- subroutine, public `mrm_read_config` (`file_namelist`, `unamelist`, `file_namelist_param`, `unamelist_param`, `do_message`, `readLatLon`)

Read the general config of mRM.
- subroutine `read_mrm_routing_params` (`processCase`, `file_namelist_param`, `unamelist_param`)

TODO: add description.

15.55.1 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

15.55.2 Function/Subroutine Documentation

15.55.2.1 `mrm_read_config()`

```
subroutine, public mo_mrm_read_config::mrm_read_config (
    character(*), intent(in) file_namelist,
    integer, intent(in) unamelist,
    character(*), intent(in) file_namelist_param,
    integer, intent(in) unamelist_param,
    logical, intent(in) do_message,
    logical, intent(out) readLatLon )
```

Read the general config of mRM.

Depending on the variable `mrm_coupling_config`, the mRM config is either read from `mrm.nml` and parameters from `mrm_parameter.nml` or copied from mHM.

Parameters

in	<code>character(*) :: file_namelist, file_namelist_param</code>	
in	<code>integer :: unamelist, unamelist_param</code>	
in	<code>character(*) :: file_namelist, file_namelist_param</code>	
in	<code>integer :: unamelist, unamelist_param</code>	
in	<code>logical :: do_message</code>	- flag for writing mHM standard messages
out	<code>logical :: readLatLon</code>	- flag for reading LatLon file

Authors

Stephan Thober

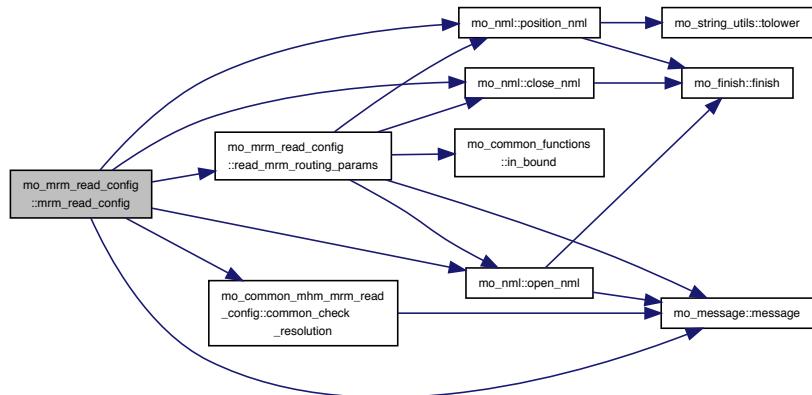
Date

Aug 2015

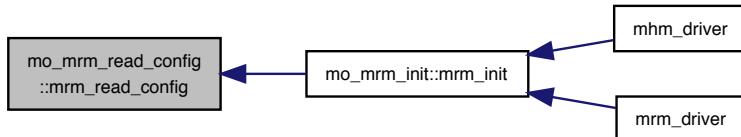
References mo_common_variables::alma_convention, mo_mrm_global_variables::basin_mrm, mo_nml::close_nml(), mo_common_mhm_mrm_read_config::common_check_resolution(), mo_mrm_global_variables::dirgauges, mo_mrm_global_variables::dirtotalrunoff, mo_mrm_file::file_defoutput, mo_mrm_global_variables::filenametotalrunoff, mo_mrm_global_variables::gauge, mo_mrm_global_variables::inflowgauge, mo_mrm_global_variables::is_start, mo_common_constants::maxnobasins, mo_mrm_constants::maxnogauges, mo_message::message(), mo_common_variables::nbasins, mo_mrm_global_variables::ngaugestotal, mo_mrm_global_variables::ninflowgauges, mo_common_constants::nodata_i4, mo_nml::open_nml(), mo_mrm_global_variables::outputflxstate_mrm, mo_nml::position_nml(), mo_common_variables::processmatrix, read_mrm_routing_params(), mo_mrm_global_variables::timestep_model_outputs_mrm, mo_mrm_file::udefoutput, and mo_mrm_global_variables::varnametotalrunoff.

Referenced by mo_mrm_init::mrm_init().

Here is the call graph for this function:



Here is the caller graph for this function:



15.55.2.2 read_mrm_routing_params()

```

subroutine mo_mrm_read_config::read_mrm_routing_params (
    integer(i4), intent(in) processCase,

```

```
character(*), intent(in) file_namelist_param,
integer(i4), intent(in) unamelist_param )
```

TODO: add description.

TODO: add description

Parameters

in	integer(i4) :: processCase	it is the default case, should be one
in	character(*) :: file_namelist_param	file name containing parameter namelist
in	integer(i4) :: unamelist_param	file name id containing parameter namelist

Authors

Robert Scheweppe

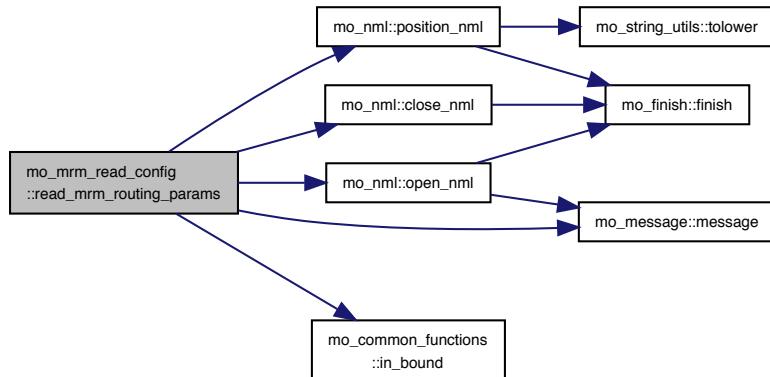
Date

Jun 2018

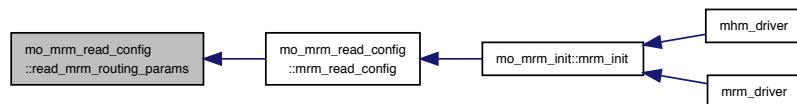
References mo_nml::close_nml(), mo_common_variables::global_parameters, mo_common_variables::global_parameters_name, mo_common_functions::in_bound(), mo_message::message(), mo_common_constants::ncolpars, mo_nml::open_nml(), mo_nml::position_nml(), and mo_common_variables::processmatrix.

Referenced by mrm_read_config().

Here is the call graph for this function:



Here is the caller graph for this function:



15.56 mo_mrm_read_data Module Reference

This module contains all routines to read mRM data from file.

Functions/Subroutines

- subroutine, public [mrm_read_l0_data](#) (do_reinit, do_readl0, 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 [rotate_fdir_variable](#) (x)
TODO: add description.

15.56.1 Detailed Description

This module contains all routines to read mRM data from file.

TODO: add description

Authors

Stephan Thober

Date

Aug 2015

15.56.2 Function/Subroutine Documentation

15.56.2.1 [mrm_read_discharge\(\)](#)

subroutine, public mo_mrm_read_data::mrm_read_discharge ()

Read discharge timeseries from file.

Read Observed discharge at the outlet of a catchment and at the inflow of a catchment. Allocate global runoff variable that contains the simulated runoff after the simulation.

Authors

Matthias Zink & Stephan Thober

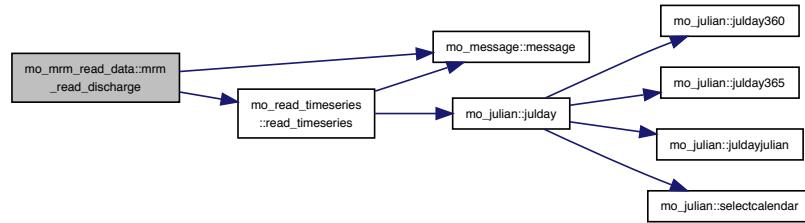
Date

Aug 2015

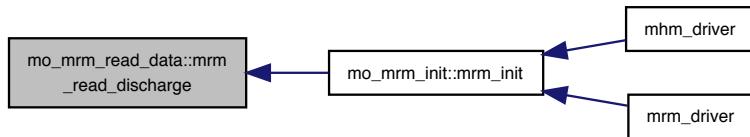
References mo_common_mhm_mrm_variables::evalper, mo_mrm_global_variables::gauge, mo_mrm_global_variables::inflowgauge, mo_message::message(), mo_mrm_global_variables::mrm_runoff, mo_common_variables::nbasins, mo_mrm_global_variables::ngaugestotal, mo_mrm_global_variables::ninfilegaugestotal, mo_mrm_global_variables::nmeasperday, mo_common_constants::nodata_dp, mo_common_mhm_mrm_variables::ntstepday, mo_common_mhm_mrm_variables::opti_function, mo_common_mhm_mrm_variables::optimize, mo_read_timeseries::read_timeseries(), mo_common_mhm_mrm_variables::simper, and mo_mrm_file::udischarge.

Referenced by `mo_mrm_init::mrm_init()`.

Here is the call graph for this function:



Here is the caller graph for this function:



15.56.2.2 mrm_read_l0_data()

```

subroutine, public mo_mrm_read_data::mrm_read_l0_data (
    logical, intent(in) do_reinit,
    logical, intent(in) do_readlatlon,
    logical, intent(in) do_readlcovr )
  
```

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	<i>logical :: do_reinit</i>	
in	<i>logical :: do_readlatlon</i>	
in	<i>logical :: do_readlcovr</i>	

Authors

Juliane Mai, Matthias Zink, and Stephan Thober

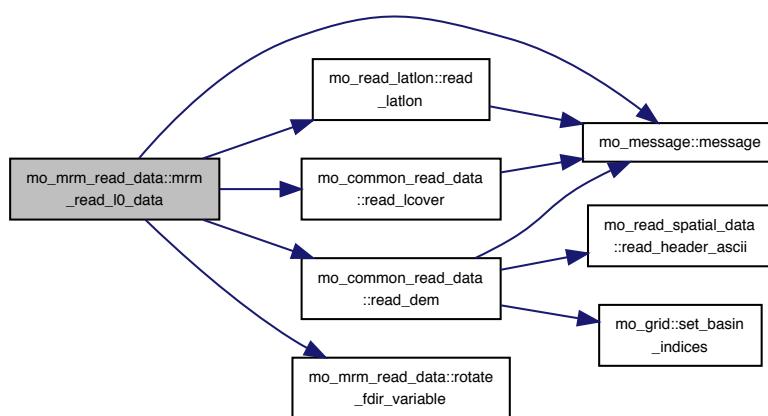
Date

Aug 2015

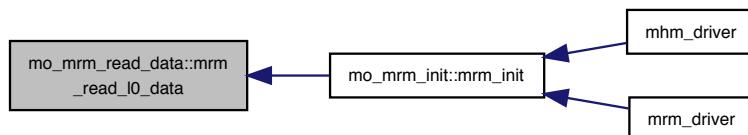
References mo_mrm_global_variables::basin_mrm, mo_common_variables::dirmorpho, mo_mrm_file::file_facc, mo_mrm_file::file_fdir, mo_mrm_file::file_gaugeloc, mo_common_variables::l0_basin, mo_mrm_global_variables::l0_facc, mo_mrm_global_variables::l0_fdir, mo_mrm_global_variables::l0_gaugeloc, mo_mrm_global_variables::l0_inflowgaugeloc, mo_common_variables::l0_lcover, mo_common_variables::level0, mo_message::message(), mo_common_variables::nbasins, mo_common_constants::nodata_i4, mo_common_variables::processmatrix, mo_common_read_data::read_dem(), mo_read_latlon::read_latlon(), mo_common_read_data::read_lcover(), rotate_fdir_variable(), mo_mrm_file::ufacc, mo_mrm_file::ufdir, and mo_mrm_file::ugaugeloc.

Referenced by mo_mrm_init::mrm_init().

Here is the call graph for this function:



Here is the caller graph for this function:



15.56.2.3 mrm_read_total_runoff()

```
subroutine, public mo_mrm_read_data::mrm_read_total_runoff (
    integer(i4), intent(in) iBasin )
```

read simulated runoff that is to be routed

read spatio-temporal field of total runoff that has been simulated by a hydrologic model or land surface model. This total runoff will then be aggregated to the level 11 resolution and then routed through the stream network.

Parameters

in	<i>integer(i4) :: iBasin</i>	basin id
----	------------------------------	----------

Authors

Stephan Thober

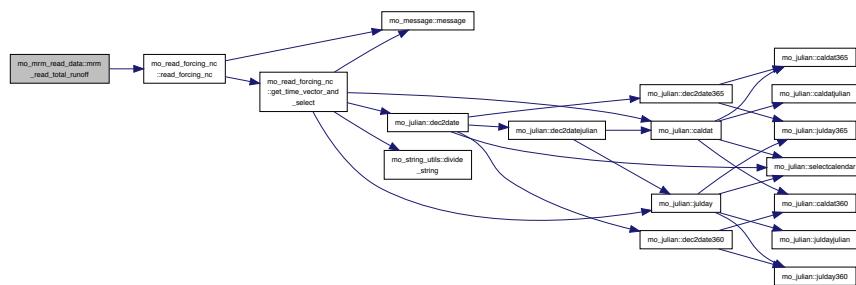
Date

Sep 2015

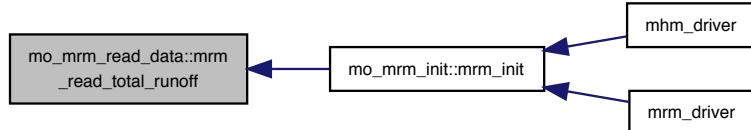
References `mo_common_variables::alma_convention`, `mo_mrm_global_variables::dirtotalrunoff`, `mo_mrm_global_variables::filenametotalrunoff`, `mo_common_constants::hoursecs`, `mo_mrm_global_variables::l1_total_runoff_in`, `mo_common_variables::level1`, `mo_common_constants::nodata_dp`, `mo_read_forcing_nc::read_forcing_nc()`, `mo_common_mhm_mrm_variables::simper`, `mo_common_mhm_mrm_variables::timestep`, and `mo_mrm_global_variables::varnametotalrunoff`.

Referenced by `mo_mrm_init::mrm_init()`.

Here is the call graph for this function:



Here is the caller graph for this function:



15.56.2.4 `rotate_fdir_variable()`

```
subroutine mo_mrm_read_data::rotate_fdir_variable (
    integer(i4), dimension(:, :), intent(inout) x )
```

TODO: add description.

TODO: add description

Parameters

in, out	integer(i4), dimension(:, :) :: x	
---------	-----------------------------------	--

Authors

L. Samaniego & R. Kumar

Date

Jun 2018

References mo_common_constants::nodata_i4.

Referenced by mrm_read_l0_data().

Here is the caller graph for this function:



15.57 mo_mrm_restart Module Reference

Restart routines.

Functions/Subroutines

- subroutine, public [mrm_write_restart](#) (iBasin, OutPath)
write routing states and configuration
- subroutine, public [mrm_read_restart_states](#) (iBasin, InPath)
read routing states
- subroutine, public [mrm_read_restart_config](#) (iBasin, InPath)
reads Level 11 configuration from a restart directory

15.57.1 Detailed Description

Restart routines.

This module contains the subroutines for reading and writing routing related variables to file.

Authors

Stephan Thober

Date

Aug 2015

15.57.2 Function/Subroutine Documentation

15.57.2.1 mrm_read_restart_config()

```
subroutine, public mo_mrm_restart::mrm_read_restart_config (
    integer(i4), intent(in) iBasin,
    character(256), intent(in) InPath )
```

reads Level 11 configuration from a restart directory

read Level 11 configuration variables from a given restart directory and initializes all Level 11 configuration variables, that are initialized in L11_variable_init, contained in module [mo_startup](#).

Parameters

in	<i>integer(i4) :: iBasin</i>	number of basin
in	<i>character(256) :: InPath</i>	Input Path including trailing slash

Authors

Stephan Thober

Date

Apr 2013

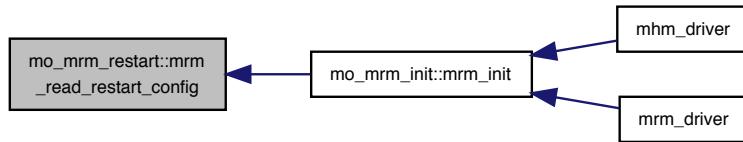
References mo_mrm_global_variables::basin_mrm, mo_kind::dp, mo_kind::i4, mo_mrm_global_variables::l11_afloodplain, mo_mrm_global_variables::l11_colout, mo_mrm_global_variables::l11_fcol, mo_mrm_global_variables::l11_fdir, mo_mrm_global_variables::l11_fromn, mo_mrm_global_variables::l11_frow, mo_mrm_global_variables::l11_l1_id, mo_mrm_global_variables::l11_label, mo_mrm_global_variables::l11_length, mo_mrm_global_variables::l11_netperm, mo_mrm_global_variables::l11_noutlets, mo_mrm_global_variables::l11_order, mo_mrm_global_variables::l11_rowout, mo_mrm_global_variables::l11_sink, mo_mrm_global_variables::l11_slope, mo_mrm_global_variables::l11_tcol, mo_mrm_global_variables::l11_ton, mo_mrm_global_variables::l11_trow, mo_mrm_global_variables::l11_tsroute, mo_mrm_global_variables::l1_l11_id, mo_common_variables::level1, mo_mrm_global_variables::level11, mo_message::message(), mo_common_variables::nbasins, mo_common_constants::nodata_dp, and mo_common_variables::processmatrix.

Referenced by [mo_mrm_init::mrm_init\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



15.57.2.2 mrm_read_restart_states()

```
subroutine, public mo_mrm_restart::mrm_read_restart_states (
    integer(i4), intent(in) iBasin,
    character(256), intent(in) InPath )
```

read routing states

This subroutine reads the routing states from mRM_states_<basin_id>.nc that has to be in the given path directory. This subroutine has to be called directly each time the mHM_eval or mRM_eval is called such that the the states are always the same at the first simulation time step, crucial for optimization.

Parameters

in	<i>integer(i4) :: iBasin</i>	number of basin
in	<i>character(256) :: InPath</i>	Input Path including trailing slash

Authors

Stephan Thober

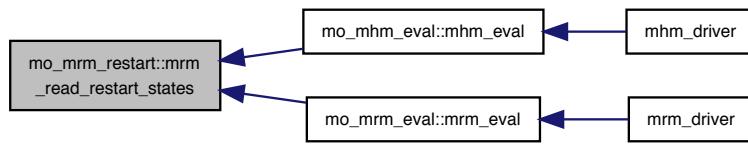
Date

Sep 2015

References `mo_mrm_global_variables::l11_c1`, `mo_mrm_global_variables::l11_c2`, `mo_mrm_global_variables::l11_k`, `mo_mrm_global_variables::l11_nlinkfracfpimp`, `mo_mrm_global_variables::l11_qmod`, `mo_mrm_global_variables::l11_qout`, `mo_mrm_global_variables::l11_qtin`, `mo_mrm_global_variables::l11_qtr`, `mo_mrm_global_variables::l11_xi`, `mo_mrm_global_variables::level11`, `mo_common_variables::nlcoverscene`, and `mo_mrm_constants::nroutingstates`.

Referenced by `mo_mhm_eval::mhm_eval()`, and `mo_mrm_eval::mrm_eval()`.

Here is the caller graph for this function:



15.57.2.3 mrm_write_restart()

```

subroutine, public mo_mrm_restart::mrm_write_restart (
    integer(i4), intent(in) iBasin,
    character(256), dimension(:), intent(in) OutPath )
write routing states and configuration
write configuration and state variables to a given restart directory.

```

Parameters

in	<code>integer(i4) :: iBasin</code>	number of basin
in	<code>character(256), dimension(:) :: OutPath</code>	list of Output paths per Basin

Authors

Stephan Thober

Date

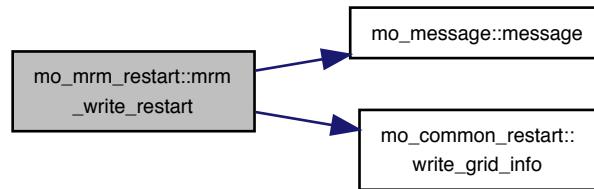
Aug 2015

References `mo_mrm_global_variables::basin_mrm`, `mo_mrm_global_variables::l11_afloodplain`, `mo_mrm_global_variables::l11_c1`, `mo_mrm_global_variables::l11_c2`, `mo_mrm_global_variables::l11_colout`, `mo_mrm_global_variables::l11_fcol`, `mo_mrm_global_variables::l11_fdir`, `mo_mrm_global_variables::l11_fromn`, `mo_mrm_global_variables::l11_frow`, `mo_mrm_global_variables::l11_k`, `mo_mrm_global_variables::l11_l1_id`, `mo_mrm_global_variables::l11_label`, `mo_mrm_global_variables::l11_length`, `mo_mrm_global_variables::l11_netperm`, `mo_mrm_global_variables::l11_nlinkfracfpimp`, `mo_mrm_global_variables::l11_qmod`, `mo_mrm_global_variables::l11_qout`, `mo_mrm_global_variables::l11_qtin`, `mo_mrm_global_variables::l11_qtr`, `mo_mrm_global_variables::l11_xi`, `mo_mrm_global_variables::level11`, `mo_common_variables::nlcoverscene`, and `mo_mrm_constants::nroutingstates`.

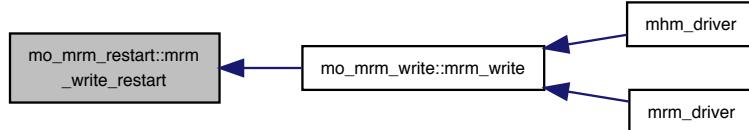
```
::l11_order, mo_mrm_global_variables::l11_rowout, mo_mrm_global_variables::l11_sink, mo_mrm_global_variables::l11_slope, mo_mrm_global_variables::l11_tcol, mo_mrm_global_variables::l11_ton, mo_mrm_global_variables::l11_trow, mo_mrm_global_variables::l11_tsrou, mo_mrm_global_variables::l11_xi, mo_mrm_global_variables::l11_id, mo_common_variables::level1, mo_mrm_global_variables::level11, mo_message::message(), mo_common_variables::nlcoverscene, mo_common_constants::nodata_dp, mo_common_constants::nodata_i4, mo_mrm_constants::nroutingstates, mo_common_variables::processmatrix, and mo_common_restart::write_grid_info().
```

Referenced by `mo_mrm_write::mrm_write()`.

Here is the call graph for this function:



Here is the caller graph for this function:



15.58 mo_mrm_routing Module Reference

Performs runoff routing for mHM at level L11.

Functions/Subroutines

- subroutine, public `mrm_routing` (read_states, processCase, global_routing_param, L1_total_runoff, L1_areaCell, L1_L11_Id, L11_areaCell, L11_L1_Id, L11_netPerm, L11_fromN, L11_toN, L11_nOutlets, timestep, tsRoutFactor, nNodes, nInflowGauges, InflowGaugeIndexList, InflowGaugeHeadwater, InflowGaugeNodeList, InflowDischarge, nGauges, gaugeIndexList, gaugeNodeList, map_flag, L11_length, L11_slope, L11_FracFPimp, L11_C1, L11_C2, L11_qOut, L11_qTIN, L11_qTR, L11_qMod, GaugeDischarge)

route water given runoff
- subroutine `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](#)).*

15.58.1 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

15.58.2 Function/Subroutine Documentation

15.58.2.1 add_inflow()

```
subroutine mo_mrm_routing::add_inflow (
    integer(i4), intent(in) nInflowGauges,
    integer(i4), dimension(:), intent(in) InflowIndexList,
    logical, dimension(:), intent(in) InflowHeadwater,
    integer(i4), dimension(:), intent(in) InflowNodeList,
    real(dp), dimension(:), intent(in) QInflow,
    real(dp), dimension(:), intent(inout) qOut )
```

Adds inflow discharge to the runoff produced at the cell where the inflow is occurring.

If a inflow gauge is given, then this routine is adding the values to the runoff produced at the grid cell where the inflow is happening. The values are not directly added to the river network. If this cell is not a headwater then the streamflow produced upstream will be neglected.

Parameters

in	integer(i4) :: nInflowGauges	[-] number of inflow points
in	integer(i4), dimension(:) :: InflowIndexList	[-] index of inflow points
in	logical, dimension(:) :: InflowHeadwater	[-] if to consider headwater cells of inflow gauge
in	integer(i4), dimension(:) :: InflowNodeList	[-] L11 ID of inflow points
in	real(dp), dimension(:) :: QInflow	[m3 s-1] inflowing water
in, out	real(dp), dimension(:) :: qOut	[m3 s-1] Series of attenuated runoff

Authors

Stephan Thober & Matthias Zink

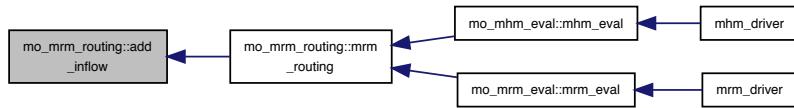
Date

Jul 2016

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

Referenced by mrm_routing().

Here is the caller graph for this function:



15.58.2.2 l11_routing()

```

subroutine mo_mrm_routing::l11_routing (
    integer(i4), intent(in) nNodes,
    integer(i4), intent(in) nLinks,
    integer(i4), dimension(:), intent(in) netPerm,
    integer(i4), dimension(:), intent(in) netLink_fromN,
    integer(i4), dimension(:), intent(in) netLink_toN,
    real(dp), dimension(:), intent(in) netLink_C1,
    real(dp), dimension(:), intent(in) netLink_C2,
    real(dp), dimension(:), intent(in) netNode_qOUT,
    integer(i4), intent(in) nInflowGauges,
    logical, dimension(:), intent(in) InflowHeadwater,
    integer(i4), dimension(:), intent(in) InflowNodeList,
    real(dp), dimension(:, :), intent(inout) netNode_qTIN,
    real(dp), dimension(:, :), intent(inout) netNode_qTR,
    real(dp), dimension(nnodes), intent(out) netNode_Qmod )

```

Performs runoff routing for mHM at L11 upscaled network ([Routing Network](#)).

Hydrograph routing is carried out with the Muskingum algorithm [7]. This simplification of the St. Venant equations is justified in mHM because the potential areas of application of this model would hardly exhibit abruptly changing hydrographs with supercritical flows. The discharge leaving the river reach located on cell i $Q_i^1(t)$ at time step t can be determined by

$$Q_i^1(t) = Q_i^1(t-1) + c_1 (Q_i^0(t-1) - Q_i^1(t-1)) + c_2 (Q_i^0(t) - Q_i^0(t-1))$$

with

$$Q_i^0(t) = Q_{i'}(t) + Q_{i'}^1(t)$$

$$c_1 = \frac{\Delta t}{\kappa(1-\xi) + \frac{\Delta t}{2}}$$

$$c_2 = \frac{\frac{\Delta t}{2} - \kappa\xi}{\kappa(1-\xi) + \frac{\Delta t}{2}}$$

where Q_i^0 and Q_i^1 denote the discharge entering and leaving the river reach located on cell i respectively. $Q_{i'}$ is the contribution from the upstream cell i' . κ Muskingum travel time parameter. ξ Muskingum attenuation parameter. Δt time interval in hours. t Time index for each Δt interval. To improve performance, a routing sequence "netPerm" is required. This permutation is determined in the mo_init_mrm routine.

TODO: add description

Parameters

in	integer(i4) :: nNodes	number of network nodes = nCells1
in	integer(i4) :: nLinks	number of stream segment (reaches)
in	integer(i4), dimension(:) :: netPerm	routing order of a given basin (permutation)
in	integer(i4), dimension(:) :: netLink_fromN	from node
in	integer(i4), dimension(:) :: netLink_toN	to node
in	real(dp), dimension(:) :: netLink_C1	routing parameter C1 ([7] p. 25-41)
in	real(dp), dimension(:) :: netLink_C2	routing parameters C2 (id)
in	real(dp), dimension(:) :: netNode_qOUT	Total outflow from cells (given basin) L11 at time tt in [m3 s-1]
in	integer(i4) :: nInflowGauges	[] number of inflow points
in	logical, dimension(:) :: InflowHeadwater	[] if to consider headwater cells of inflow gauge
in	integer(i4), dimension(:) :: InflowNodeList	[] L11 ID of inflow points
in, out	real(dp), dimension(:, :) :: netNode_qTIN	[m3 s-1] Total inputs at t-1 and t
in, out	real(dp), dimension(:, :) :: netNode_qTR	[m3 s-1] Transformed outflow leaving node l (Muskingum)
out	real(dp), dimension(nNodes) :: netNode_Qmod	[m3 s-1] Simulated routed discharge

Authors

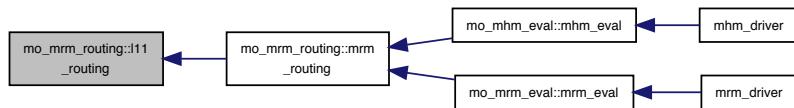
Luis Samaniego

Date

Dec 2005

Referenced by mrm_routing().

Here is the caller graph for this function:



15.58.2.3 l11_runoff_acc()

```

subroutine mo_mrm_routing::l11_runoff_acc (
    real(dp), dimension(:), intent(in) qAll,
    real(dp), dimension(:), intent(in) efecArea,
    integer(i4), dimension(:), intent(in) L1_L11_Id,
    real(dp), dimension(:), intent(in) L11_areaCell,
    integer(i4), dimension(:), intent(in) L11_L1_Id,
    integer(i4), intent(in) TS,
    logical, intent(in) map_flag,
    real(dp), dimension(:), intent(out) qAcc )
  
```

total runoff accumulation at L11.

Upscales runoff in space from L1 to L11 if routing resolution is higher than hydrology resolution (map_flag equals .true.) or downscals runoff from L1 to L11 if routing resolution is lower than hydrology resolution.

Parameters

in	<i>real(dp), dimension(:) :: qall</i>	total runoff L1 [mm tst-1]
in	<i>real(dp), dimension(:) :: efecarea</i>	effective area in [km ²] at Level 1
in	<i>integer(i4), dimension(:) :: L1_L11_Id</i>	L11 lds mapped on L1
in	<i>real(dp), dimension(:) :: L11_areacell</i>	effective area in [km ²] at Level 11
in	<i>integer(i4), dimension(:) :: L11_L1_Id</i>	L1 lds mapped on L11
in	<i>integer(i4) :: TS</i>	time step in [s]
in	<i>logical :: map_flag</i>	Flag indicating whether routing resolution is higher than hydrologic one
out	<i>real(dp), dimension(:) :: qAcc</i>	aggregated runoff at L11 [m ³ s ⁻¹]

Authors

Luis Samaniego

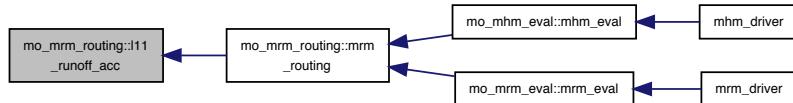
Date

Jan 2013

References mo_common_constants::hoursecs, and mo_common_constants::nodata_dp.

Referenced by mrm_routing().

Here is the caller graph for this function:



15.58.2.4 mrm_routing()

```

subroutine, public mo_mrm_routing:::mrm_routing (
    logical, intent(in) read_states,
    integer(i4), intent(in) processCase,
    real(dp), dimension(:), intent(in) global_routing_param,
    real(dp), dimension(:), intent(in) L1_total_runoff,
    real(dp), dimension(:), intent(in) L1_areacell,
    integer(i4), dimension(:), intent(in) L1_L11_Id,
    real(dp), dimension(:), intent(in) L11_areacell,
    integer(i4), dimension(:), intent(in) L11_L1_Id,
    integer(i4), dimension(:), intent(in) L11_netPerm,
    integer(i4), dimension(:), intent(in) L11_fromN,
  
```

```

integer(i4), dimension(:), intent(in) L11_toN,
integer(i4), intent(in) L11_nOutlets,
integer(i4), intent(in) timestep,
real(dp), intent(in) tsRoutFactor,
integer(i4), intent(in) nNodes,
integer(i4), intent(in) nInflowGauges,
integer(i4), dimension(:), intent(in) InflowGaugeIndexList,
logical, dimension(:), intent(in) InflowGaugeHeadwater,
integer(i4), dimension(:), intent(in) InflowGaugeNodeList,
real(dp), dimension(:), intent(in) InflowDischarge,
integer(i4), intent(in) nGauges,
integer(i4), dimension(:), intent(in) gaugeIndexList,
integer(i4), dimension(:), intent(in) gaugeNodeList,
logical, intent(in) map_flag,
real(dp), dimension(:), intent(in) L11_length,
real(dp), dimension(:), intent(in) L11_slope,
real(dp), dimension(:), intent(in) L11_FracFPimp,
real(dp), dimension(:), intent(inout) L11_C1,
real(dp), dimension(:), intent(inout) L11_C2,
real(dp), dimension(:), intent(inout) L11_qOut,
real(dp), dimension(:, :), intent(inout) L11_qTIN,
real(dp), dimension(:, :), intent(inout) L11_qTR,
real(dp), dimension(:, :), intent(inout) L11_qMod,
real(dp), dimension(:, :), intent(inout) GaugeDischarge )

```

route water given runoff

This routine first performs mpr for the routing variables if required, then accumulates the runoff to the routing resolution and eventually routes the water in a third step. The last step is repeated multiple times if the routing timestep is smaller than the timestep of the hydrological timestep

Parameters

in	<i>logical</i> :: <i>read_states</i>	whether states are derived from restart file
in	<i>integer(i4)</i> :: <i>processCase</i>	Process switch for routing
in	<i>real(dp)</i> , dimension(:) :: <i>global_routing_param</i>	routing parameters
in	<i>real(dp)</i> , dimension(:) :: <i>L1_total_runoff</i>	total runoff from L1 grid cells
in	<i>real(dp)</i> , dimension(:) :: <i>L1_areaCell</i>	L1 cell area
in	<i>integer(i4)</i> , dimension(:) :: <i>L1_L11_Id</i>	L1 cell ids on L11
in	<i>real(dp)</i> , dimension(:) :: <i>L11_areaCell</i>	L11 cell area
in	<i>integer(i4)</i> , dimension(:) :: <i>L11_L1_Id</i>	L11 cell ids on L1
in	<i>integer(i4)</i> , dimension(:) :: <i>L11_netPerm</i>	L11 routing order
in	<i>integer(i4)</i> , dimension(:) :: <i>L11_fromN</i>	L11 source grid cell order
in	<i>integer(i4)</i> , dimension(:) :: <i>L11_toN</i>	L11 target grid cell order
in	<i>integer(i4)</i> :: <i>L11_nOutlets</i>	L11 number of outlets/sinks
in	<i>integer(i4)</i> :: <i>timestep</i>	simulation timestep in [h]
in	<i>real(dp)</i> :: <i>tsRoutFactor</i>	factor between routing timestep and hydrological timestep
in	<i>integer(i4)</i> :: <i>nNodes</i>	number of nodes
in	<i>integer(i4)</i> :: <i>nInflowGauges</i>	number of inflow gauges
in	<i>integer(i4)</i> , dimension(:) :: <i>InflowGaugeIndexList</i>	index list of inflow gauges
in	<i>logical</i> , dimension(:) :: <i>InflowGaugeHeadwater</i>	flag for headwater cell of inflow gauge
in	<i>integer(i4)</i> , dimension(:) :: <i>InflowGaugeNodeList</i>	gauge node list at L11

Parameters

in	<i>real(dp), dimension(:) :: InflowDischarge</i>	inflowing discharge at discharge gauge at current day
in	<i>integer(i4) :: nGauges</i>	number of recording gauges
in	<i>integer(i4), dimension(:) :: gaugeIndexList</i>	index list for outflow gauges
in	<i>integer(i4), dimension(:) :: gaugeNodeList</i>	gauge node list at L11
in	<i>logical :: map_flag</i>	flag indicating whether routing resolution is coarser than hydrologic resolution
in	<i>real(dp), dimension(:) :: L11_length</i>	L11 link length
in	<i>real(dp), dimension(:) :: L11_slope</i>	L11 slope
in	<i>real(dp), dimension(:) :: L11_FracFPimp</i>	L11 fraction of flood plain with impervious cover
in, out	<i>real(dp), dimension(:) :: L11_C1</i>	L11 muskingum parameter 1
in, out	<i>real(dp), dimension(:) :: L11_C2</i>	L11 muskingum parameter 2
in, out	<i>real(dp), dimension(:) :: L11_qOut</i>	total runoff from L11 grid cells
in, out	<i>real(dp), dimension(:, :) :: L11_qTIN</i>	L11 inflow to the reach
in, out	<i>real(dp), dimension(:, :) :: L11_qTR</i>	L11 routed outflow
in, out	<i>real(dp), dimension(:) :: L11_qMod</i>	modelled discharge at each grid cell
in, out	<i>real(dp), dimension(:) :: GaugeDischarge</i>	modelled discharge at each gauge

Authors

Stephan Thober

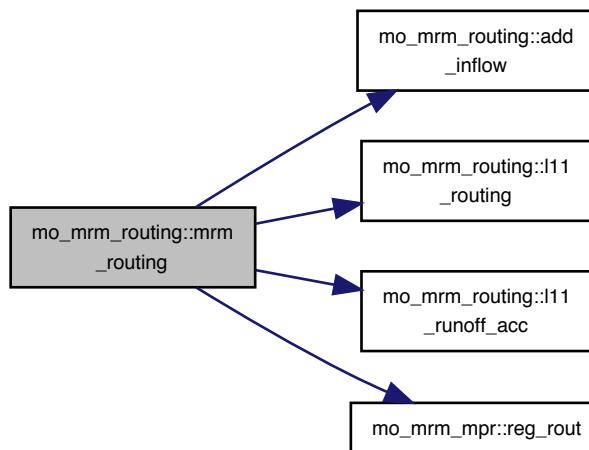
Date

Aug 2015

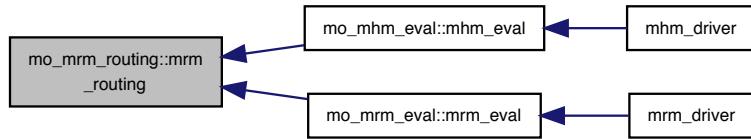
References `add_inflow()`, `mo_mrm_global_variables::is_start`, `l11_routing()`, `l11_runoff_acc()`, and `mo_mrm_mpr::reg_rout()`.

Referenced by `mo_mhm_eval::mhm_eval()`, and `mo_mrm_eval::mrm_eval()`.

Here is the call graph for this function:



Here is the caller graph for this function:



15.59 mo_mrm_signatures Module Reference

Module with calculations for several hydrological signatures.

Functions/Subroutines

- `real(dp) function, dimension(size(lags, 1)), public autocorrelation (data, lags, mask)`
Autocorrelation of a given data series.
- `real(dp) function, dimension(size(quantiles, 1)), public flowdurationcurve (data, quantiles, mask, concavity_index, mid_segment_slope, mhigh_segment_volume, high_segment_volume, low_segment_volume)`
Flow duration curves.
- subroutine, public `limb_densities` (data, mask, RLD, DLD)
Calculates limb densities.
- `real(dp) function maximummonthlyflow (data, mask, yr_start, mo_start, dy_start)`
Maximum of average flows per months.
- subroutine, public `moments` (data, mask, mean_data, stddev_data, median_data, max_data, mean_log, stddev_log, median_log, max_log)
Moments of data and log-transformed data, e.g. mean and standard deviation.
- `real(dp) function, dimension(size(quantiles, 1)), public peakdistribution (data, quantiles, mask, slope_peak_distribution)`
Calculates the peak distribution.
- `real(dp) function, public runoffratio (data, basin_area, mask, precip_series, precip_sum, log_data)`
Runoff ratio (accumulated daily discharge [mm/d] / accumulated daily precipitation [mm/d]).
- `real(dp) function, public zeroflowratio (data, mask)`
Ratio of zero values to total number of data points.

15.59.1 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

15.59.2 Function/Subroutine Documentation**15.59.2.1 autocorrelation()**

```
real(dp) function, dimension(size(lags, 1)), public mo_mrm_signatures::autocorrelation (
    real(dp), dimension(:), intent(in) data,
    integer(i4), dimension(:), intent(in) lags,
    logical, dimension(size(data, 1)), intent(in), optional mask )
```

Autocorrelation of a given data series.

Calculates the autocorrelation of a data series at given lags. An optional argument for masking data points can be given. The function is basically a wrapper of the function autocorr from the module [mo_corr](#). An optional mask of data points can be specified. ADDITIONAL INFORMATION Used as hydrologic signature with lag 1 in Euser, T., Winsemius, H. C., Hrachowitz, M., Fenicia, F., Uhlenbrook, S., & Savenije, H. H. G. (2013). A framework to assess the realism of model structures using hydrological signatures. *Hydrology and Earth System Sciences*, 17(5), 1893-1912. doi:10.5194/hess-17-1893-2013

Parameters

in	<i>real(dp), dimension(:) :: data</i>	Array of data
in	<i>integer(i4), dimension(:) :: lags</i>	Array of lags where autocorrelation is requested
in	<i>logical, dimension(size(data, 1)), optional :: mask</i>	Mask for data points given. Works only with 1d double precision input data.

Authors

Juliane Mai

Date

Jun 2015

15.59.2.2 flowdurationcurve()

```
real(dp) function, dimension(size(quantiles, 1)), public mo_mrm_signatures::flowdurationcurve (
    real(dp), dimension(:), intent(in) data,
    real(dp), dimension(:), intent(in) quantiles,
    logical, dimension(:), intent(in), optional mask,
    real(dp), intent(out), optional concavity_index,
    real(dp), intent(out), optional mid_segment_slope,
    real(dp), intent(out), optional mhigh_segment_volume,
    real(dp), intent(out), optional high_segment_volume,
    real(dp), intent(out), optional low_segment_volume )
```

Flow duration curves.

Calculates the flow duration curves for a given data vector. The Flow duration curve at a certain quantile x is the data point p where $x\%$ of the data points are above the value p . Hence the function percentile of the module `mo_percentile` is used. But percentile is determining the point p where $x\%$ of the data points are below that value. Therefore, 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 $h = 1, 2, \dots, H$ 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 $h = 1, 2, \dots, H$ 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 $l = 1, 2, \dots, L$ are flow indeces located within the low-flow segment (exceedance probabilities larger than m_1). L is the index of the minimum flow. The settings $m_1 = 0.7$ is used by Shafii et. al (2014) and is implemented like that. An optional mask of data points can be specified. ADDITIONAL INFORMATION Thresholds in `mid_segment_slope`, `mhigh_segment_volume`, `high_segment_volume`, `low_segment_volume` are hard coded. FDC is used as hydrologic signature (quantiles not specified) in Euser, T., Winsemius, H. C., Hrachowitz, M., Fenicia, F., Uhlenbrook, S., & Savenije, H. H. G. (2013). A framework to assess the realism of model structures using hydrological signatures. *Hydrology and Earth System Sciences*, 17(5), 1893-1912. doi:10.5194/hess-17-1893-2013 Concavity Index used as hydrologic signature in Zhang, Y., Vaze, J., Chiew, F. H. S., Teng, J., & Li, M. (2014). Predicting hydrological signatures in ungauged catchments using spatial interpolation, index model, and rainfall-runoff modelling. *Journal of Hydrology*, 517(C), 936-948. doi:10.1016/j.jhydrol.2014.06.032 Concavity index is defined using exceedance probabilities by Sauquet, E., & Catalogne, C. (2011). Comparison of catchment grouping methods for flow duration curve estimation at ungauged sites in France. *Hydrology and Earth System Sciences*, 15(8), 2421-2435. doi:10.5194/hess-15-2421-2011 `mid_segment_slope`, `high_segment_volume`, `low_segment_volume` used as hydrologic signature in Shafii, M., & Tolson, B. A. (2015). Optimizing hydrological consistency by incorporating hydrological signatures into model calibration objectives. *Water Resources Research*, 51(5), 3796-3814. doi:10.1002/2014WR016520

Parameters

in	<code>real(dp)</code> , <code>dimension(:)</code> :: data	data series
in	<code>real(dp)</code> , <code>dimension(:)</code> :: quantiles	Percentages of exceedance (x-axis of FDC)
in	<code>logical</code> , <code>dimension(:)</code> , <code>optional</code> :: mask	mask of data array
out	<code>real(dp)</code> , <code>optional</code> :: concavity_index	concavity index as defined by Sauquet et al. (2011)

Parameters

out	<i>real(dp), optional :: mid_segment_slope</i>	mid-segment slope as defined by Shafii et al. (2014)
out	<i>real(dp), optional :: mhigh_segment_volume</i>	medium high-segment volume
out	<i>real(dp), optional :: high_segment_volume</i>	high-segment volume as defined by Shafii et al. (2014)
out	<i>real(dp), optional :: low_segment_volume</i>	low-segment volume as defined by Shafii et al. (2014)

Returns

real(dp), dimension(size(quantiles,1)) :: FlowDurationCurve — Flow Duration Curve value at resp. quantile

Authors

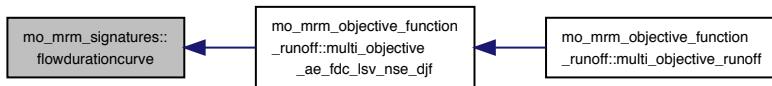
Remko Nijzink, Juliane Mai

Date

March 2014

Referenced by `mo_mrm_objective_function_runoff::multi_objective_ae_fdc_lsv_nse_djf()`.

Here is the caller graph for this function:



15.59.2.3 limb_densities()

```

subroutine, public mo_mrm_signatures::limb_densities (
    real(dp), dimension(:), intent(in) data,
    logical, dimension(size(data, 1)), intent(in), optional mask,
    real(dp), intent(out), optional RLD,
    real(dp), intent(out), optional DLD )

```

Calculates limb densities.

Calculates rising and declining 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	<i>real(dp), dimension(:) :: data</i>	data series
in	<i>logical, dimension(size(data, 1)), optional :: mask</i>	mask for data series
out	<i>real(dp), optional :: RLD</i>	rising limb density
out	<i>real(dp), optional :: DLD</i>	declining limb density

Authors

Remko Nijzink

Date

March 2014

References `mo_message::message()`.

Here is the call graph for this function:



15.59.2.4 maximummonthlyflow()

```
real(dp) function mo_mrm_signatures::maximummonthlyflow (
    real(dp), dimension(:), intent(in) data,
    logical, dimension(size(data, 1)), intent(in), optional mask,
    integer(i4), intent(in), optional yr_start,
    integer(i4), intent(in), optional mo_start,
    integer(i4), intent(in), optional dy_start )
```

Maximum of average flows per months.

Maximum of average flow per month is defined as

$$max_{monthlyflow} = \text{Max}(F(i), i = 1..12)$$

where $\$F(i)$ 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	<i>real(dp), dimension(:) :: data</i>	array of data
in	<i>logical, dimension(size(data, 1)), optional :: mask</i>	mask for data points given
in	<i>integer(i4), optional :: yr_start</i>	year of date of first data point given
in	<i>integer(i4), optional :: mo_start</i>	month of date of first data point given (default: 1)
in	<i>integer(i4), optional :: dy_start</i>	month of date of first data point given (default: 1)

Returns

`real(dp) :: MaximumMonthlyFlow` — Maximum of average flow per month Works only with 1d double precision input data. Assumes data are daily values.

Authors

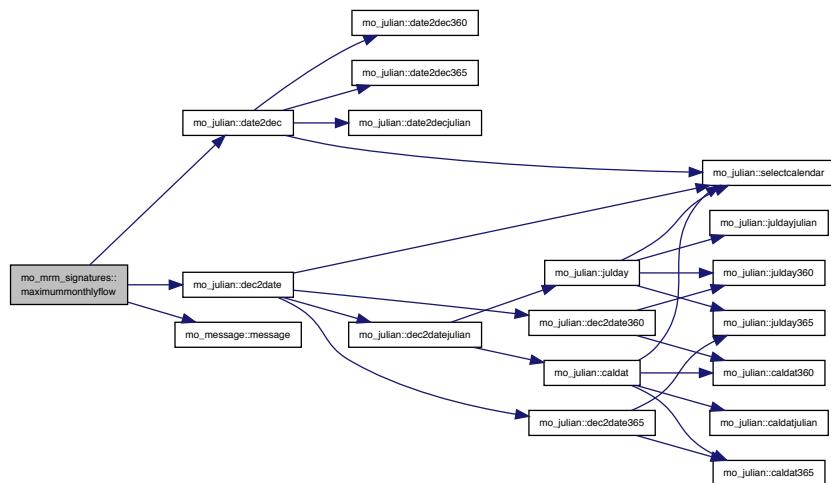
Juliane Mai

Date

Jun 2015

References `mo_julian::date2dec()`, `mo_julian::dec2date()`, and `mo_message::message()`.

Here is the call graph for this function:

**15.59.2.5 moments()**

```
subroutine, public mo_mrm_signatures::moments (
    real(dp), dimension(:, intent(in) data,
    logical, dimension(size(data, 1)), intent(in), optional mask,
    real(dp), intent(out), optional mean_data,
    real(dp), intent(out), optional stddev_data,
    real(dp), intent(out), optional median_data,
    real(dp), intent(out), optional max_data,
    real(dp), intent(out), optional mean_log,
    real(dp), intent(out), optional stddev_log,
    real(dp), intent(out), optional median_log,
    real(dp), intent(out), optional max_log )
```

Moments of data and log-transformed data, e.g. mean and standard deviation.

Returns several moments of data series given, i.e.

- mean of data

- standard deviation of data
- median of data
- maximum/ peak of data
- mean of log-transformed data
- standard deviation of log-transformed data
- median of log-transformed data
- maximum/ peak of log-transformed data An optional mask of data points can be specified. ADDITIONAL INFORMATION mean_log and stddev_log used as hydrologic signature in Zhang, Y., Vaze, J., Chiew, F. H. S., Teng, J., & Li, M. (2014). Predicting hydrological signatures in ungauged catchments using spatial interpolation, index model, and rainfall-runoff modelling. *Journal of Hydrology*, 517(C), 936-948. doi:10.1016/j.jhydrol.2014.06.032 mean_data, stddev_data, median_data, max_data, mean_log, and stddev_log used as hydrologic signature in Shafii, M., & Tolson, B. A. (2015). Optimizing hydrological consistency by incorporating hydrological signatures into model calibration objectives. *Water Resources Research*, 51(5), 3796-3814. doi:10.1002/2014WR016520

Parameters

in	<i>real(dp), dimension(:) :: data</i>	array of data
in	<i>logical, dimension(size(data, 1)), optional :: mask</i>	mask for data points given
out	<i>real(dp), optional :: mean_data</i>	mean of data
out	<i>real(dp), optional :: stddev_data</i>	standard deviation of data
out	<i>real(dp), optional :: median_data</i>	median of data
out	<i>real(dp), optional :: max_data</i>	maximum/ peak of data
out	<i>real(dp), optional :: mean_log</i>	mean of log-transformed data
out	<i>real(dp), optional :: stddev_log</i>	standard deviation of log-transformed data
out	<i>real(dp), optional :: median_log</i>	median of log-transformed data
out	<i>real(dp), optional :: max_log</i>	maximum/ peak of log-transformed data Works only with 1d double precision input data.

Authors

Juliane Mai

Date

Jun 2015

References `mo_message::message()`.

Here is the call graph for this function:



15.59.2.6 peakdistribution()

```
real(dp) function, dimension(size(quantiles, 1)), public mo_mrm_signatures::peakdistribution (
    real(dp), dimension(:, intent(in) data,
    real(dp), dimension(:, intent(in) quantiles,
    logical, dimension(size(data, 1)), intent(in), optional mask,
    real(dp), intent(out), optional slope_peak_distribution )
```

Calculates the peak distribution.

First, the peaks of the time series given are identified. For the peak distribution only this subset of data points are considered. Second, the peak distribution at the quantiles given is calculated. Calculates the peak distribution at the quantiles given using [mo_percentile](#). Since the exceedance probabilities are usually used in hydrology the function percentile is used with (1.0-quantiles). Optionally, the slope of the peak distribution between 10th and 50th percentile, i.e.

$$slope = \frac{\text{peak_data}_{0.1} - \text{peak_data}_{0.5}}{0.9 - 0.5}$$

can be returned. An optional mask for the data points can be given. ADDITIONAL INFORMATION slope_peak_distribution used as hydrologic signature in Euser, T., Winsemius, H. C., Hrachowitz, M., Fenicia, F., Uhlenbrook, S., & Savenije, H. H. G. (2013). A framework to assess the realism of model structures using hydrological signatures. *Hydrology and Earth System Sciences*, 17(5), 1893-1912. doi:10.5194/hess-17-1893-2013

Parameters

in	<i>real(dp), dimension(:) :: data</i>	data array
in	<i>real(dp), dimension(:) :: quantiles</i>	requested quantiles for distribution
in	<i>logical, dimension(size(data, 1)), optional :: mask</i>	mask of data array
out	<i>real(dp), optional :: slope_peak_distribution</i>	slope of the Peak distribution between 10th and 50th percentile

Returns

real(dp), dimension(size(quantiles,1)) :: PeakDistribution — Distribution of peak values at resp. quantiles

Authors

Remko Nijzink

Date

March 2014

15.59.2.7 runoffratio()

```
real(dp) function, public mo_mrm_signatures::runoffratio (
    real(dp), dimension(:, intent(in) data,
    real(dp), intent(in) basin_area,
    logical, dimension(size(data, 1)), intent(in), optional mask,
    real(dp), dimension(size(data, 1)), intent(in), optional precip_series,
    real(dp), intent(in), optional precip_sum,
    logical, intent(in), optional log_data )
```

Runoff ratio (accumulated daily discharge [mm/d] / accumulated daily precipitation [mm/d]).

The runoff ratio is defined as

$$\text{runoffratio} = \frac{\sum_{t=1}^N q_t}{\sum_{t=1}^N p_t}$$

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

$$\text{runoff_ratio} = \frac{\sum_{t=1}^N \log(q_t)}{\sum_{t=1}^N p_t}$$

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	<code>real(dp), dimension(:) :: data</code>	array of data [m**3/s]
in	<code>real(dp) :: basin_area</code>	area of basin [km**2]
in	<code>logical, dimension(size(data, 1)), optional :: mask</code>	mask for data points given
in	<code>real(dp), dimension(size(data, 1)), optional :: precip_series</code>	daily precipitation values [mm/km**2 / day]
in	<code>real(dp), optional :: precip_sum</code>	sum of daily precip. values of whole period[mm/km**2 / day]
in	<code>logical, optional :: log_data</code>	ratio using logarithmic dataWorks only with 1d double precision input data.

Authors

Juliane Mai

Date

Jun 2015

References `mo_message::message()`.

Here is the call graph for this function:



15.59.2.8 zeroflowratio()

```
real(dp) function, public mo_mrm_signatures::zeroflowratio (
    real(dp), dimension(:), intent(in) data,
    logical, dimension(size(data, 1)), intent(in), optional mask )
```

Ratio of zero values to total number of data points.

An optional mask of data points can be specified. ADDITIONAL INFORMATION

Returns

real(dp), dimension(size(lags,1)) :: ZeroFlowRatio — Ratio of zero values to total number of data points Used as hydrologic signature in Zhang, Y., Vaze, J., Chiew, F. H. S., Teng, J., & Li, M. (2014). Predicting hydrological signatures in ungauged catchments using spatial interpolation, index model, and rainfall-runoff modelling. *Journal of Hydrology*, 517(C), 936-948. doi:10.1016/j.jhydrol.2014.06.032

Parameters

in	<i>real(dp), dimension(:) :: data</i>	array of data
in	<i>logical, dimension(size(data, 1)), optional :: mask</i>	mask for data points givenWorks only with 1d double precision input data.

Authors

Juliane Mai

Date

Jun 2015

References *mo_message::message()*.

Here is the call graph for this function:



15.60 mo_mrm_write Module Reference

write of discharge and restart files

Functions/Subroutines

- subroutine, public [mrm_write](#)
write discharge and restart files
- subroutine [write_configfile](#)

This module 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.

Variables

- integer(i4) `day_counter`
- integer(i4) `month_counter`
- integer(i4) `year_counter`
- integer(i4) `average_counter`
- type(outputdataset) `nc`

15.60.1 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

15.60.2 Function/Subroutine Documentation

15.60.2.1 mrm_write()

subroutine, public mo_mrm_write:::mrm_write ()

write discharge and restart files

First, this subroutine calls the writing or restart files that only succeeds if it happens after the write of mHM restart files because mHM restart files must exist. Second, simulated discharge is aggregated to the daily scale and then written to file jointly with observed discharge

Authors

Juliane Mai, Rohini Kumar & Stephan Thober

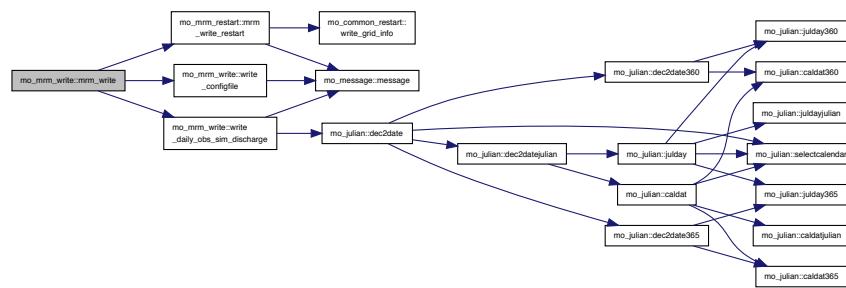
Date

Aug 2015

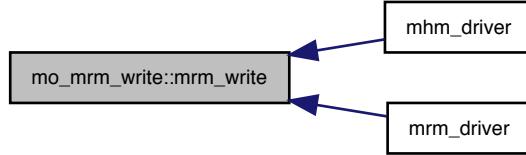
References mo_mrm_global_variables::basin_mrm, mo_common_variables::dirrestartout, mo_common_mhm_mrm_variables::evalper, mo_mrm_global_variables::gauge, mo_common_mhm_mrm_variables::mrm_coupling_mode, mo_mrm_global_variables::mrm_runoff, mo_mrm_restart::mrm_write_restart(), mo_common_variables::nbasins, mo_mrm_global_variables::ngaugestotal, mo_common_mhm_mrm_variables::ntstepday, mo_common_mhm_mrm_variables::simper, mo_common_mhm_mrm_variables::warmingdays, write_configfile(), write_daily_obs_sim_discharge(), and mo_common_variables::write_restart.

Referenced by mhm_driver(), and mrm_driver().

Here is the call graph for this function:



Here is the caller graph for this function:



15.60.2.2 mrm_write_optifile()

```
subroutine, public mo_mrm_write::mrm_write_optifile (
    real(dp), intent(in) best_OF,
    real(dp), dimension(:), intent(in) best_paramSet,
    character(len = *), dimension(:), intent(in) param_names )
```

Write briefly final optimization results.

Write overall best objective function and the best optimized parameter set to a file_opti.

Parameters

in	<i>real(dp) :: best_OF</i>	best objective function value as returned by the optimization routine
in	<i>real(dp), dimension(:) :: best_paramSet</i>	best associated global parameter set Called only when optimize is .TRUE.
in	<i>character(len = *), dimension(:) :: param_names</i>	

Authors

David Schaefer

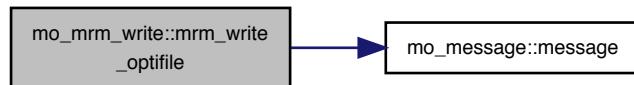
Date

July 2013

References mo_common_variables::dirconfigout, mo_common_mhm_mrm_file::file_opti, mo_message::message(), and mo_common_mhm_mrm_file::uopti.

Referenced by mrm_driver().

Here is the call graph for this function:



Here is the caller graph for this function:



15.60.2.3 mrm_write_optinamelist()

```

subroutine, public mo_mrm_write::mrm_write_optinamelist (
    real(dp), dimension(:, :), intent(in) parameters,
    logical, dimension(size(parameters, 1)), intent(in) maskpara,
    character(len = *), dimension(size(parameters, 1)), intent(in) parameters_name )

```

Write final, optimized parameter set in a namelist format.

Write final, optimized parameter set in a namelist format.

Parameters

in	<i>real(dp), dimension(:, :) :: parameters</i>	(min, max, opti)
in	<i>logical, dimension(size(parameters, 1)) :: maskpara</i>	.true. if parameter was calibrated
in	<i>character(len = *), dimension(size(parameters, 1)) :: parameters_name</i>	clear names of parameters

Authors

Juliane Mai

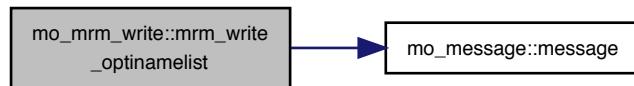
Date

Dec 2013

References mo_common_variables::dirconfigout, mo_common_mhm_mrm_file::file_opti_nml, mo_message::message(), mo_common_variables::processmatrix, and mo_common_mhm_mrm_file::uopti_nml.

Referenced by mrm_driver().

Here is the call graph for this function:



Here is the caller graph for this function:



15.60.2.4 mrm_write_output_fluxes()

```

subroutine, public mo_mrm_write::mrm_write_output_fluxes (
    integer(i4), intent(in) iBasin,
    integer(i4), intent(in) nCells,
    integer(i4), intent(in) timeStep_model_outputs,
    integer(i4), intent(in) warmingDays,
    real(dp), intent(in) newTime,
    integer(i4), intent(in) nTimeSteps,
  
```

```

integer(i4), intent(in) nTStepDay,
integer(i4), intent(in) tt,
integer(i4), intent(in) day,
integer(i4), intent(in) month,
integer(i4), intent(in) year,
integer(i4), intent(in) timestep,
logical, dimension(:, :), intent(in) 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	<i>integer(i4) :: iBasin</i>	
in	<i>integer(i4) :: nCells</i>	
in	<i>integer(i4) :: timeStep_model_outputs</i>	timestep of model outputs
in	<i>integer(i4) :: warmingDays</i>	number of warming days
in	<i>real(dp) :: newTime</i>	julian date of next time step
in	<i>integer(i4) :: nTimeSteps</i>	number of total timesteps
in	<i>integer(i4) :: nTStepDay</i>	number of timesteps per day
in	<i>integer(i4) :: tt</i>	current model timestep
in	<i>integer(i4) :: day</i>	current day of the year
in	<i>integer(i4) :: month</i>	current month of the year
in	<i>integer(i4) :: year</i>	current year
in	<i>integer(i4) :: timestep</i>	current model time resolution
in	<i>logical, dimension(:, :) :: mask11</i>	mask at level 11
in	<i>real(dp), dimension(:) :: L11_qMod</i>	current routed streamflow

Authors

Stephan Thober

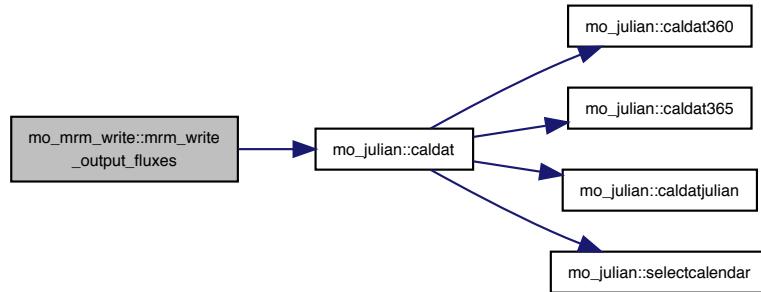
Date

Aug 2015

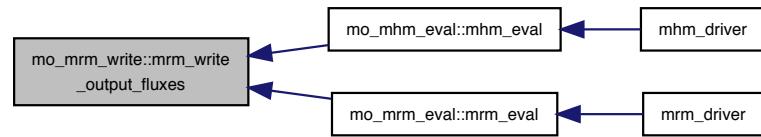
References average_counter, mo_julian::caldat(), day_counter, mo_kind::dp, mo_kind::i4, month_counter, nc, and year_counter.

Referenced by mo_mhm_eval::mhm_eval(), and mo_mrm_eval::mrm_eval().

Here is the call graph for this function:



Here is the caller graph for this function:



15.60.2.5 write_configfile()

```
subroutine mo_mrm_write::write_configfile ( )
```

This module writes the results of the configuration into an ASCII-file.

TODO: add description

Authors

Christoph Schneider

Date

May 2013

References mo_mrm_global_variables::basin_mrm, mo_common_variables::dirconfigout, mo_mrm_global_variables::dirgauges, mo_common_variables::dirlcover, mo_common_variables::dirmorpho, mo_common_variables::dirout, mo_common_variables::dirrestartout, mo_mrm_global_variables::dirtotalrunoff, mo_kind::dp, mo_common_mhm_mrm_variables::evalper, mo_common_file::file_config, mo_mrm_global_variables::gauge, mo_common_variables::global_parameters, mo_common_variables::global_parameters_name, mo_kind::i4, mo_mrm_global_variables::inflowgauge, mo_common_variables::i0_basin, mo_mrm_global_variables::i11_fromn, mo_mrm_global_variables::i11_i1_id, mo_mrm_global_variables::i11_label, mo_mrm_global_variables::i11_length, mo_mrm_global_variables::i11_netperm, mo_mrm_global_variables::i11_rorder, mo_mrm::

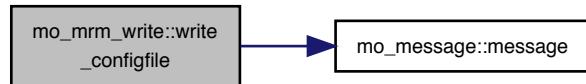
```

_global_variables::l11_slope, mo_mrm_global_variables::l11_ton, mo_mrm_global_variables::l1_l11_id, mo_
_common_variables::lc_year_end, mo_common_variables::lc_year_start, mo_common_variables::lcfilename,
mo_common_mhm_mrm_variables::lcyearid, mo_common_variables::level0, mo_common_variables::level1,
mo_mrm_global_variables::level11, mo_message::message(), mo_common_mhm_mrm_variables::mrm_
coupling_mode, mo_common_variables::nbasins, mo_mrm_global_variables::ngaugestotal, mo_mrm_global_
variables::ninfowaugestotal, mo_common_variables::nlcoverscene, mo_common_constants::nodata_dp, mo_
_common_variables::processmatrix, mo_common_mhm_mrm_variables::read_restart, mo_common_variables_
::resolutionhydrology, mo_common_mhm_mrm_variables::resolutionrouting, mo_common_mhm_mrm_variables_
::simper, mo_common_mhm_mrm_variables::timestep, mo_common_file::uconfig, mo_mrm_file::version, mo_
common_mhm_mrm_variables::warmper, and mo_common_variables::write_restart.

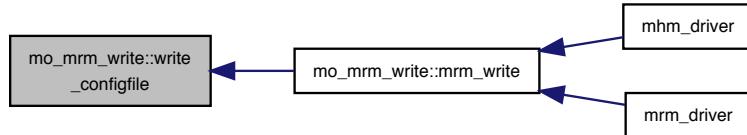
```

Referenced by mrm_write().

Here is the call graph for this function:



Here is the caller graph for this function:



15.60.2.6 write_daily_obs_sim_discharge()

```

subroutine mo_mrm_write:::write_daily_obs_sim_discharge (
    real(dp), dimension(:, :, ), intent(in) Qobs,
    real(dp), dimension(:, :, ), intent(in) Qsim )

```

Write a file for the daily observed and simulated discharge timeseries during the evaluation period for each gauging station.

Write a file for the daily observed and simulated discharge timeseries during the evaluation period for each gauging station

Parameters

in	real(dp), dimension(:, :,) :: Qobs	daily time series of observed discharge dims = (nModeling_days , nGauges_total)
----	-------------------------------------	--

Parameters

in	<code>real(dp), dimension(:, :) :: Qsim</code>	daily time series of modeled dischargedims = (nModeling_days , nGauges_total)
----	--	---

Authors

Rohini Kumar

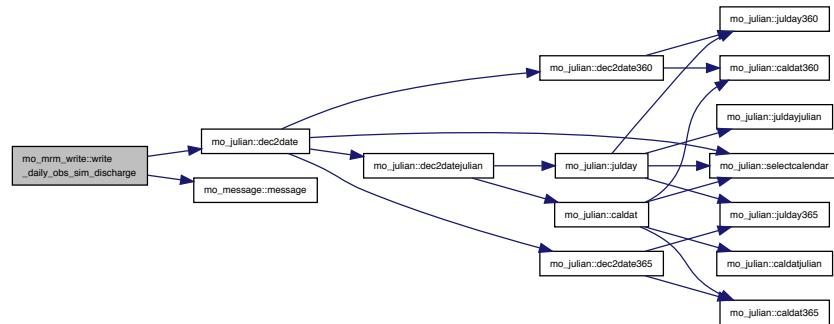
Date

August 2013

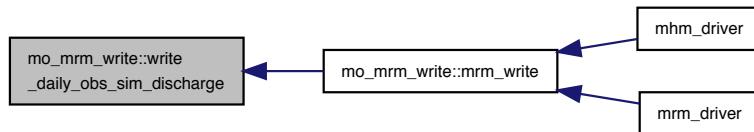
References `mo_mrm_global_variables::basin_mrm`, `mo_julian::dec2date()`, `mo_common_variables::dirout`, `mo_common_mhm_mrm_variables::evalper`, `mo_mrm_file::file_daily_discharge`, `mo_mrm_global_variables::gauge`, `mo_message::message()`, `mo_common_variables::nbasins`, `mo_mrm_file::ncfile_discharge`, and `mo_mrm_file::udaily_discharge`.

Referenced by `mrm_write()`.

Here is the call graph for this function:



Here is the caller graph for this function:



15.60.3 Variable Documentation

15.60.3.1 average_counter

```
integer(i4) mo_mrm_write::average_counter [private]
```

Referenced by `mrm_write_output_fluxes()`.

15.60.3.2 day_counter

```
integer(i4) mo_mrm_write::day_counter [private]
```

Referenced by `mrm_write_output_fluxes()`.

15.60.3.3 month_counter

```
integer(i4) mo_mrm_write::month_counter [private]
```

Referenced by `mrm_write_output_fluxes()`.

15.60.3.4 nc

```
type(outputdataset) mo_mrm_write::nc [private]
```

Referenced by `mrm_write_output_fluxes()`.

15.60.3.5 year_counter

```
integer(i4) mo_mrm_write::year_counter [private]
```

Referenced by `mrm_write_output_fluxes()`.

15.61 mo_mrm_write_fluxes_states Module Reference

Creates NetCDF output for different fluxes and state variables of mHM.

Data Types

- interface [outputdataset](#)
- interface [outputvariable](#)

Functions/Subroutines

- type([outputvariable](#)) function `newoutputvariable` (nc, name, dtype, dims, ncells, mask, avg)
Initialize OutputVariable.
- subroutine `updatevariable` (self, data)
Update OutputVariable.
- subroutine `writevariabletimestep` (self, timestep)
Write timestep to file.

- type([outputdataset](#)) function [newoutputdataset](#) (ibasin, mask, nCells)
Initialize OutputDataset.
- subroutine [updatedataset](#) (self, sidx, eidx, L11_Qmod)
Update all variables.
- subroutine [writetimestep](#) (self, timestep)
Write all accumulated data.
- subroutine [close](#) (self)
Close the file.
- type(ncdataset) function [createoutputfile](#) (ibasin)
Create and initialize output file.
- subroutine [writevariableattributes](#) (var, long_name, unit)
Write output variable attributes.

15.61.1 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

15.61.2 Function/Subroutine Documentation

15.61.2.1 [close\(\)](#)

```
subroutine mo_mrm_write_fluxes_states::close (
    class(outputdataset) self ) [private]
```

Close the file.

Close the file associated with variable of type(OutputDataset)

Authors

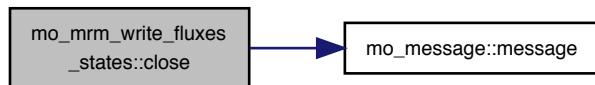
Rohini Kumar & Stephan Thober

Date

August 2013

References `mo_common_variables::dirout`, and `mo_message::message()`.

Here is the call graph for this function:



15.61.2.2 `createoutputfile()`

```
type(ncdataset) function mo_mrm_write_fluxes_states::createoutputfile (
    integer(i4), intent(in) ibasin )
```

Create and initialize output file.

Create output file, write all non-dynamic variables and global attributes for the given basin.

Returns

`type(NcDataset)`

Parameters

in	<code>integer(i4) :: ibasin</code>	-> basin id
----	------------------------------------	-------------

Authors

David Schaefer

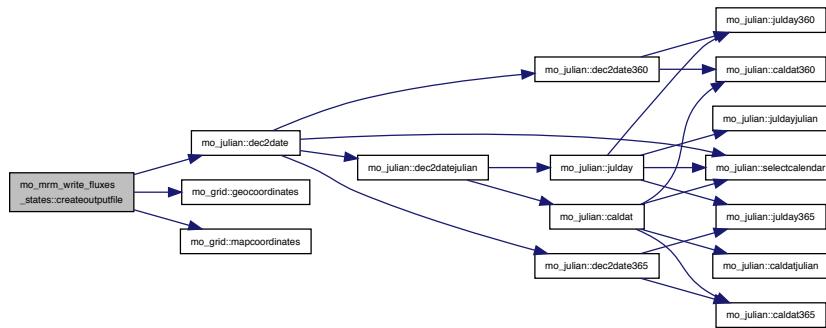
Date

June 2015

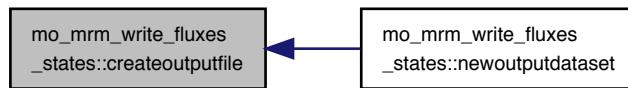
References `mo_julian::dec2date()`, `mo_common_variables::dirout`, `mo_common_mhm_mrm_variables::evalper`, `mo_mrm_file::file_mrm_output`, `mo_grid::geocoordinates()`, `mo_mrm_global_variables::level11`, `mo_grid::mapcoordinates()`, `mo_common_constants::nodata_dp`, and `mo_mrm_file::version`.

Referenced by `newoutputdataset()`.

Here is the call graph for this function:



Here is the caller graph for this function:



15.61.2.3 newoutputdataset()

```

type(outputdataset) function mo_mrm_write_fluxes_states::newoutputdataset (
    integer(i4), intent(in) ibasin,
    logical, dimension(:, :, ), intent(in), target 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	<i>integer(i4) :: ibasin</i>	-> basin id
in	<i>logical, dimension(:, :) :: mask</i>	
in	<i>integer(i4) :: nCells</i>	

Authors

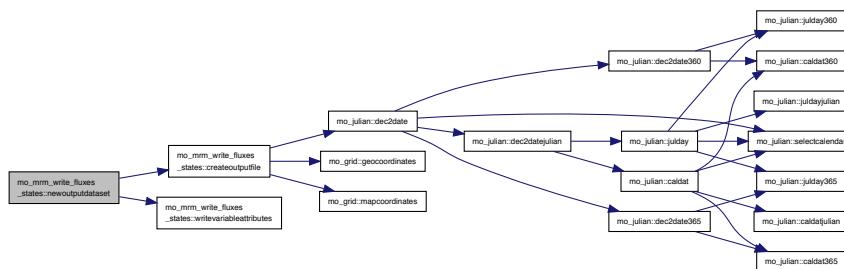
Matthias Zink

Date

Apr 2013

References `createoutputfile()`, `mo_mrm_global_variables::outputflxstate_mrm`, and `writevariableattributes()`.

Here is the call graph for this function:



15.61.2.4 newoutputvariable()

```

type(outputvariable) function mo_mrm_write_fluxes_states::newoutputvariable (
    type(ncdataset), intent(in) nc,
    character(*), intent(in) name,
    character(*), intent(in) dtype,
    character(16), dimension(3), intent(in) dims,
    integer(i4), intent(in) ncells,
    logical, dimension(:, :, ), intent(in), target mask,
    logical, intent(in), optional avg ) [private]

```

Initialize OutputVariable.

TODO: add description

Returns

`type(OutputVariable)`

Parameters

in	<i>type(NcDataset) :: nc</i>	-> NcDataset which contains the variable
in	<i>character(*) :: name</i>	
in	<i>character(*) :: dtype</i>	

Parameters

in	<i>character(16), dimension(3) :: dims</i>	
in	<i>integer(i4) :: ncells</i>	-> number of cells in basin
in	<i>logical, dimension(:, :) :: mask</i>	
in	<i>logical, optional :: avg</i>	-> average the data before writing

Authors

David Schaefer

Date

June 2015

15.61.2.5 updatedataset()

```
subroutine mo_mrm_write_fluxes_states::updatedataset (
    class(outputdataset), intent(inout), target self,
    integer(i4), intent(in) sidx,
    integer(i4), intent(in) eidx,
    real(dp), dimension(:), intent(in) L11_Qmod )
```

Update all variables.

Call the type bound procedure updateVariable for all output variables. If a new output variable needs to be written, this is the second of two procedures to change (first: newOutputDataset)

Parameters

in, out	<i>class(OutputDataset) :: self</i>	
in	<i>integer(i4) :: sidx, eidx</i>	- start index of the basin related data in L1_* arguments
in	<i>integer(i4) :: sidx, eidx</i>	- end index of the basin related data in L1_* arguments
in	<i>real(dp), dimension(:) :: L11_Qmod</i>	

Authors

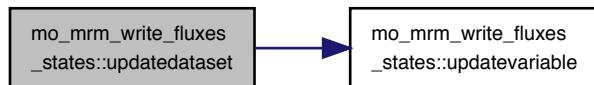
Matthias Zink

Date

Apr 2013

References `mo_mrm_global_variables::outputflxstate_mrm`, and `updatevariable()`.

Here is the call graph for this function:



15.61.2.6 updatevariable()

```
subroutine mo_mrm_write_fluxes_states::updatevariable (
    class(outputvariable), intent(inout) self,
    real(dp), dimension(:), intent(in) data ) [private]
```

Update OutputVariable.

Add the array given as actual argument to the derived type's component 'data'

Returns

`type(OutputVariable)`

Parameters

in, out	<code>class(OutputVariable) :: self</code>	
in	<code>real(dp), dimension(:) :: data</code>	

Authors

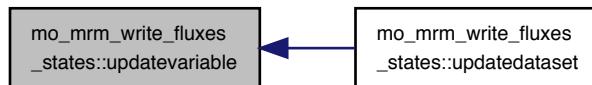
David Schaefer

Date

June 2015

Referenced by `updatedataset()`.

Here is the caller graph for this function:

**15.61.2.7 writetimestep()**

```
subroutine mo_mrm_write_fluxes_states::writetimestep (
    class(outputdataset), intent(inout), target self,
    integer(i4), intent(in) timestep )
```

Write all accumulated data.

Write all accumulated and potentially averaged data to disk.

Returns

`type(OutputVariable)`

Parameters

in, out	<code>class(OutputDataset) :: self</code>	
in	<code>integer(i4) :: timestep</code>	The model timestep to write

Authors

David Schaefer

Date

June 2015

15.61.2.8 writevariableattributes()

```
subroutine mo_mrm_write_fluxes_states::writevariableattributes (
    type(outputvariable), intent(in) var,
    character(*), intent(in) long_name,
    character(*), intent(in) unit )
```

Write output variable attributes.

TODO: add description

Parameters

in	<i>type(OutputVariable)</i> :: var	
in	<i>character(*)</i> :: <i>long_name, unit</i>	-> variable name
in	<i>character(*)</i> :: <i>long_name, unit</i>	-> physical unit

Authors

David Schaefer

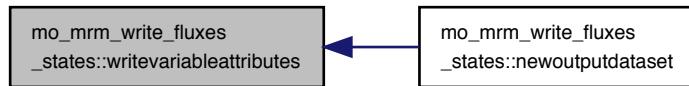
Date

June 2015

References `mo_common_constants::nodata_dp`.

Referenced by `newoutputdataset()`.

Here is the caller graph for this function:



15.61.2.9 writevariabletimestep()

```
subroutine mo_mrm_write_fluxes_states::writevariabletimestep (
    class(outputvariable), intent(inout) self,
    integer(i4), intent(in) timestep ) [private]
```

Write timestep to file.

Write the content of the derived types's component 'data' to file, average if necessary

Parameters

in, out	<i>class(OutputVariable)</i> :: <i>self</i>	
in	<i>integer(i4)</i> :: <i>timestep</i>	-> index along the time dimension of the netcdf variable

Authors

David Schaefer

Date

June 2015

References mo_common_constants::nodata_dp.

15.62 mo_multi_param_reg Module Reference

Multiscale parameter regionalization (MPR).

Functions/Subroutines

- subroutine, public `mpo` (mask0, geoUnit0, soilld0, Asp0, gridded_LAI0, LCover0, slope_emp0, y0, Id0, upper_bound1, lower_bound1, left_bound1, right_bound1, n_subcells1, fSealed1, alpha1, degDayInc1, degDayMax1, degDayNoPre1, fAsp1, HarSamCoeff1, PrieTayAlpha1, aeroResist1, surfResist1, fRoots1, kFastFlow1, kSlowFlow1, kBaseFlow1, kPerco1, karstLoss1, soilMoistFC1, soilMoistSat1, soilMoistExp1, jarvis_thresh_c1, tempThresh1, unsatThresh1, sealedThresh1, wiltingPoint1, maxInter1, petLAIcorFactor, parameterset)

Regionalizing and Upscaling process parameters.
- subroutine `baseflow_param` (param, geoUnit0, k2_0)

baseflow recession parameter
- subroutine `snow_acc_melt_param` (param, c2TSTu, fForest1, flperm1, fPerm1, tempThresh1, degDayNoPre1, degDayInc1, degDayMax1)

Calculates the snow parameters.
- subroutine `iper_thres_runoff` (param, sealedThresh1)

sets the impervious layer threshold parameter for runoff generation
- subroutine `karstic_layer` (param, geoUnit0, mask0, SMs_FC0, KsVar_V0, Id0, n_subcells1, upper_bound1, lower_bound1, left_bound1, right_bound1, karstLoss1, L1_Kp)

calculates the Karstic percolation loss
- subroutine, public `canopy_intercept_param` (processMatrix, param, LAI0, n_subcells1, upper_bound1, lower_bound1, left_bound1, right_bound1, Id0, mask0, nodata, max_intercept1)

estimate effective maximum interception capacity at L1
- subroutine `aerodynamical_resistance` (LAI0, LCover0, param, mask0, Id0, n_subcells1, upper_bound1, lower_bound1, left_bound1, right_bound1, aerodyn_resistance1)

Regionalization of aerodynamic resistance.

15.62.1 Detailed Description

Multiscale parameter regionalization (MPR).

This module provides the routines for multiscale parameter regionalization (MPR).

Authors

Stephan Thober, Rohini Kumar

Date

Dec 2012

15.62.2 Function/Subroutine Documentation

15.62.2.1 aerodynamical_resistance()

```
subroutine mo_multi_param_reg::aerodynamical_resistance (
    real(dp), dimension(:, :, ), intent(in) LAI0,
    integer(i4), dimension(:, ), intent(in) LCover0,
    real(dp), dimension(6), intent(in) param,
    logical, dimension(:, :, ), intent(in) mask0,
    integer(i4), dimension(:, ), intent(in) Id0,
    integer(i4), dimension(:, ), intent(in) n_subcells1,
    integer(i4), dimension(:, ), intent(in) upper_bound1,
    integer(i4), dimension(:, ), intent(in) lower_bound1,
    integer(i4), dimension(:, ), intent(in) left_bound1,
    integer(i4), dimension(:, ), intent(in) right_bound1,
    real(dp), dimension(:, :, ), intent(out) aerodyn_resistance1 )
```

Regionalization of aerodynamic resistance.

estimation of aerodynamical resistance Global parameters needed (see mhm_parameter.nml):

- param(1) = canopyheight_forest
- param(2) = canopyheight_impermeous
- param(3) = canopyheight_pervious
- param(4) = displacementheight_coeff
- param(5) = roughnesslength_momentum_coeff
- param(6) = roughnesslength_heat_coeff

Parameters

in	<i>real(dp), dimension(:, :,) :: LAI0</i>	LAI at level-0
in	<i>integer(i4), dimension(:,) :: LCover0</i>	land cover field
in	<i>real(dp), dimension(6) :: param</i>	input parameter
in	<i>logical, dimension(:, :,) :: mask0</i>	mask at level 0
in	<i>integer(i4), dimension(:,) :: Id0</i>	Cell ids of hi res field
in	<i>integer(i4), dimension(:,) :: n_subcells1</i>	number of l0 cells within a l1 cell
in	<i>integer(i4), dimension(:,) :: upper_bound1</i>	upper row of a l1 cell in l0 grid
in	<i>integer(i4), dimension(:,) :: lower_bound1</i>	lower row of a l1 cell in l0 grid
in	<i>integer(i4), dimension(:,) :: left_bound1</i>	left col of a l1 cell in l0 grid
in	<i>integer(i4), dimension(:,) :: right_bound1</i>	right col of a l1 cell in l0 grid
out	<i>real(dp), dimension(:, :,) :: aerodyn_resistance1</i>	aerodynamic resistance

Authors

Matthias Zink

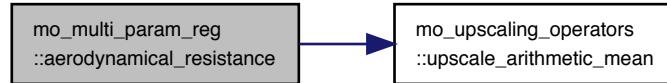
Date

Apr 2013

References `mo_common_constants::eps_dp`, `mo_mpr_constants::karman`, `mo_upscaling_operators::upscale_\leftarrowarithmetic_mean()`, and `mo_mpr_constants::windmeasheight`.

Referenced by `mpr()`.

Here is the call graph for this function:



Here is the caller graph for this function:



15.62.2.2 baseflow_param()

```

subroutine mo_multi_param_reg::baseflow_param (
    real(dp), dimension(:), intent(in) param,
    integer(i4), dimension(:), intent(in) geoUnit0,
    real(dp), dimension(:,), intent(out) k2_0 )

```

baseflow recession parameter

This subroutine calculates the baseflow recession parameter based on the geological units at the Level 0 scale. For each level 0 cell, it assigns the value specified in the parameter array param for the geological unit in this cell. Global parameters needed (see mhm_parameter.nml):

- param(1) = GeoParam(1,:)
- param(2) = GeoParam(2,:)
- ...

Parameters

in	<i>real(dp), dimension(:) :: param</i>	list of required parameters
in	<i>integer(i4), dimension(:) :: geoUnit0</i>	ids of geological units at L0
out	<i>real(dp), dimension(:,) :: k2_0</i>	- baseflow recession parameter at Level 0

Authors

Stephan Thober, Rohini Kumar

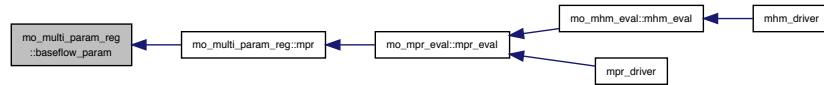
Date

Dec 2012

References mo_mpr_global_variables::geounitlist, and mo_common_constants::nodata_dp.

Referenced by mpr().

Here is the caller graph for this function:



15.62.2.3 canopy_intercept_param()

```

subroutine, public mo_multi_param_reg::canopy_intercept_param (
    integer(i4), dimension(:, :, ), intent(in) processMatrix,
    real(dp), dimension(:, ), intent(in) param,
    real(dp), dimension(:, :, ), intent(in) LAI0,
    integer(i4), dimension(:, ), intent(in) n_subcells1,
    integer(i4), dimension(:, ), intent(in) upper_bound1,
    integer(i4), dimension(:, ), intent(in) lower_bound1,
    integer(i4), dimension(:, ), intent(in) left_bound1,
    integer(i4), dimension(:, ), intent(in) right_bound1,
    integer(i4), dimension(:, ), intent(in) Id0,
    logical, dimension(:, :, ), intent(in) mask0,
    real(dp), intent(in) nodata,
    real(dp), dimension(:, :, ), intent(out) max_intercept1 )

```

estimate effective maximum interception capacity at L1

estimate effective maximum interception capacity at L1 for a given Leaf Area Index field. Global parameters needed (see mhm_parameter.nml): Process Case 1:

- param(1) = canopyInterceptionFactor

Parameters

in	<i>integer(i4), dimension(:, :,) :: processMatrix</i>	indicate processes
in	<i>real(dp), dimension(:,) :: param</i>	array of global parameters
in	<i>real(dp), dimension(:, :,) :: LAI0</i>	LAI at level-0(nCells0, time)
in	<i>integer(i4), dimension(:,) :: n_subcells1</i>	Number of L0 cells within a L1 cell
in	<i>integer(i4), dimension(:,) :: upper_bound1</i>	Upper row of high resolution block
in	<i>integer(i4), dimension(:,) :: lower_bound1</i>	Lower row of high resolution block
in	<i>integer(i4), dimension(:,) :: left_bound1</i>	Left column of high resolution block
in	<i>integer(i4), dimension(:,) :: right_bound1</i>	Right column of high resolution block
in	<i>integer(i4), dimension(:,) :: Id0</i>	Cell ids at level 0
in	<i>logical, dimension(:, :,) :: mask0</i>	mask at level 0 field
in	<i>real(dp) :: nodata</i>	- nodata value
out	<i>real(dp), dimension(:, :,) :: max_intercept1</i>	max interception at level-1(nCells1, time)

Authors

Rohini Kumar

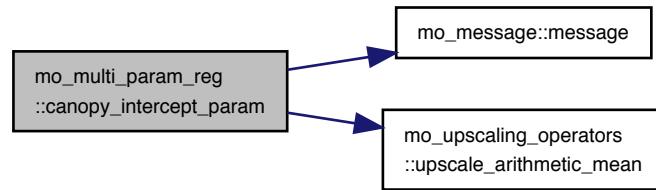
Date

Aug. 2013

References mo_message::message(), and mo_upscaling_operators::upscale_arithmetic_mean().

Referenced by mpr().

Here is the call graph for this function:



Here is the caller graph for this function:



15.62.2.4 iper_thres_runoff()

```

subroutine mo_multi_param_reg::iper_thres_runoff (
    real(dp), dimension(1), intent(in) param,
    real(dp), dimension(:, :, :), intent(out) sealedThresh1 ) [private]
  
```

sets the impervious layer threshold parameter for runoff generation

to be done by Kumar Global parameters needed (see mhm_parameter.nml):

- param(1) = imperviousStorageCapacity

Parameters

in	real(dp), dimension(1) :: param	- given threshold parameter
out	real(dp), dimension(:, :, :) :: sealedThresh1	

Authors

Stephan Thober, Rohini Kumar

Date

Dec 2012

Referenced by mpr().

Here is the caller graph for this function:



15.62.2.5 karstic_layer()

```

subroutine mo_multi_param_reg::karstic_layer (
    real(dp), dimension(3), intent(in) param,
    integer(i4), dimension(:), intent(in) geoUnit0,
    logical, dimension(:, :), intent(in) mask0,
    real(dp), dimension(:), intent(in) SMS_FCO,
    real(dp), dimension(:), intent(in) KsVar_V0,
    integer(i4), dimension(:), intent(in) Id0,
    integer(i4), dimension(:), intent(in) n_subcells1,
    integer(i4), dimension(:), intent(in) upper_bound1,
    integer(i4), dimension(:), intent(in) lower_bound1,
    integer(i4), dimension(:), intent(in) left_bound1,
    integer(i4), dimension(:), intent(in) right_bound1,
    real(dp), dimension(:), intent(out) karstLoss1,
    real(dp), dimension(:), intent(out) L1_Kp ) [private]

```

calculates the Karstic percolation loss

This subroutine calls first the karstic_fraction upscaling routine for determine the karstic fraction area for every Level 1 cell. Then, the karstic percolation loss is estimated given two shape parameters by

$$karstLoss1 = 1 + (fKarArea * param(1)) * ((-1) ** INT(param(2), i4))$$

where *karstLoss1* is the karstic percolation loss and *fKarArea* is the fraction of karstic area at level 1 Global parameters needed (see mhm_parameter.nml):

- param(1) = rechargeCoefficient
- param(2) = rechargeFactor_karstic
- param(3) = gain_loss_GWreservoir_karstic

Parameters

in	<i>real(dp), dimension(3) :: param</i>	parameters
in	<i>integer(i4), dimension(:) :: geoUnit0</i>	id of the Karstic formation
in	<i>logical, dimension(:, :) :: mask0</i>	mask at level 0

Parameters

in	<i>real(dp), dimension(:) :: SMs_FC0</i>	[‐] soil moisture deficit from field capacity w.r.t to saturation
in	<i>real(dp), dimension(:) :: KsVar_V0</i>	[‐] relative variability of saturated
in	<i>integer(i4), dimension(:) :: Id0</i>	Cell ids of hi res field
in	<i>integer(i4), dimension(:) :: n_subcells1</i>	number of l0 cells within a l1 cell
in	<i>integer(i4), dimension(:) :: upper_bound1</i>	upper row of a l1 cell in l0 grid
in	<i>integer(i4), dimension(:) :: lower_bound1</i>	lower row of a l1 cell in l0 grid
in	<i>integer(i4), dimension(:) :: left_bound1</i>	left col of a l1 cell in l0 grid
in	<i>integer(i4), dimension(:) :: right_bound1</i>	right col of a l1 cell in l0 grid
out	<i>real(dp), dimension(:) :: karstLoss1</i>	[‐] Karstic percolation loss
out	<i>real(dp), dimension(:) :: L1_Kp</i>	[d‐1] percolation coefficient

Authors

Rohini Kumar, Stephan Thober

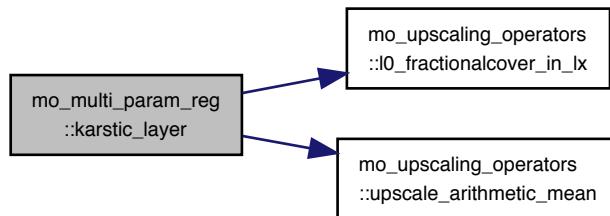
Date

Feb 2013

References mo_mpr_global_variables::c2tstu, mo_mpr_global_variables::geounitkar, mo_mpr_global_variables::geounitlist, mo_upscaling_operators::l0_fractionalcover_in_lx(), mo_common_constants::nodata_dp, mo_common_constants::nodata_i4, and mo_upscaling_operators::upscale_arithmetic_mean().

Referenced by mpr().

Here is the call graph for this function:



Here is the caller graph for this function:



15.62.2.6 mpr()

```
subroutine, public mo_multi_param_reg::mpr (
    logical, dimension(:, :, ), intent(in) mask0,
    integer(i4), dimension(:, ), intent(in) geoUnit0,
    integer(i4), dimension(:, :, ), intent(in) soilId0,
    real(dp), dimension(:, ), intent(in) Asp0,
    real(dp), dimension(:, :, ), intent(in) gridded_LAI0,
    integer(i4), dimension(:, :, ), intent(in) LCover0,
    real(dp), dimension(:, ), intent(in) slope_emp0,
    real(dp), dimension(:, ), intent(in) y0,
    integer(i4), dimension(:, ), intent(in) Id0,
    integer(i4), dimension(:, ), intent(in) upper_bound1,
    integer(i4), dimension(:, ), intent(in) lower_bound1,
    integer(i4), dimension(:, ), intent(in) left_bound1,
    integer(i4), dimension(:, ), intent(in) right_bound1,
    integer(i4), dimension(:, ), intent(in) n_subcells1,
    real(dp), dimension(:, :, :, ), intent(inout) fSealed1,
    real(dp), dimension(:, :, :, ), intent(inout) alphal,
    real(dp), dimension(:, :, :, ), intent(inout) degDayIncl,
    real(dp), dimension(:, :, :, ), intent(inout) degDayMax1,
    real(dp), dimension(:, :, :, ), intent(inout) degDayNoPre1,
    real(dp), dimension(:, :, :, ), intent(inout) fAsp1,
    real(dp), dimension(:, :, :, ), intent(inout) HarSamCoeff1,
    real(dp), dimension(:, :, :, ), intent(inout) PrieTayAlphal,
    real(dp), dimension(:, :, :, ), intent(inout) aeroResist1,
    real(dp), dimension(:, :, :, ), intent(inout) surfResist1,
    real(dp), dimension(:, :, :, ), intent(inout) fRoots1,
    real(dp), dimension(:, :, :, ), intent(inout) kFastFlow1,
    real(dp), dimension(:, :, :, ), intent(inout) kSlowFlow1,
    real(dp), dimension(:, :, :, ), intent(inout) kBaseFlow1,
    real(dp), dimension(:, :, :, ), intent(inout) kPercol,
    real(dp), dimension(:, :, :, ), intent(inout) karstLoss1,
    real(dp), dimension(:, :, :, ), intent(inout) soilMoistFC1,
    real(dp), dimension(:, :, :, ), intent(inout) soilMoistSat1,
    real(dp), dimension(:, :, :, ), intent(inout) soilMoistExpl,
    real(dp), dimension(:, :, :, ), intent(inout) jarvis_thresh_c1,
    real(dp), dimension(:, :, :, ), intent(inout) tempThresh1,
    real(dp), dimension(:, :, :, ), intent(inout) unsatThresh1,
    real(dp), dimension(:, :, :, ), intent(inout) sealedThresh1,
    real(dp), dimension(:, :, :, ), intent(inout) wiltingPoint1,
    real(dp), dimension(:, :, :, ), intent(inout) maxInter1,
    real(dp), dimension(:, :, :, ), intent(inout) petLAIcorFactor,
    real(dp), dimension(:, ), intent(in), optional, target parameterset )

```

Regionalizing and Upscaling process parameters.

calculating process parameters at L0 scale (Regionalization), like:

- Baseflow recession parameter
- Soil moisture parameters
- PET correction for aspect and upscale these parameters to retrieve effective parameters at scale L1. Further parameter regionalizations are done for:
 - snow accumulation and melting parameters
 - threshold parameter for runoff generation on impervious layer
 - karstic percolation loss

- setting up the Regionalized Routing Parameters

Parameters

in	<i>logical, dimension(:, :) :: mask0</i>	mask at level 0 field
in	<i>integer(i4), dimension(:) :: geoUnit0</i>	L0 geological units
in	<i>integer(i4), dimension(:, :) :: soilId0</i>	soil Ids at level 0
in	<i>real(dp), dimension(:) :: Asp0</i>	[degree] Aspect at Level 0
in	<i>real(dp), dimension(:, :) :: gridded_LAI0</i>	LAI grid at level 0, with dim2 = time
in	<i>integer(i4), dimension(:, :) :: LCOVER0</i>	land cover at level 0
in	<i>real(dp), dimension(:) :: slope_emp0</i>	Empirical quantiles of slope
in	<i>real(dp), dimension(:) :: y0</i>	y0 at level 0
in	<i>integer(i4), dimension(:) :: Id0</i>	Cell ids at level 0
in	<i>integer(i4), dimension(:) :: upper_bound1</i>	Upper row of hi res block
in	<i>integer(i4), dimension(:) :: lower_bound1</i>	Lower row of hi res block
in	<i>integer(i4), dimension(:) :: left_bound1</i>	Left column of hi res block
in	<i>integer(i4), dimension(:) :: right_bound1</i>	Right column of hi res block
in	<i>integer(i4), dimension(:) :: n_subcells1</i>	Number of L0 cells within a L1 cell
in, out	<i>real(dp), dimension(:, :, :) :: fSealed1</i>	[1] fraction of sealed area
in, out	<i>real(dp), dimension(:, :, :) :: alpha1</i>	[1] Exponent for the upper reservoir
in, out	<i>real(dp), dimension(:, :, :) :: degDayInc1</i>	[d-1 degreeC-1] Increase of the Degree-day factor per mm of increase in precipitation
in, out	<i>real(dp), dimension(:, :, :) :: degDayMax1</i>	[mm-1 degreeC-1] Maximum Degree-day factor
in, out	<i>real(dp), dimension(:, :, :) :: degDayNoPre1</i>	[mm-1 degreeC-1] Degree-day factor with no precipitation
in, out	<i>real(dp), dimension(:, :, :) :: fAsp1</i>	[1] PET correction for Aspect at level 1
in, out	<i>real(dp), dimension(:, :, :) :: HarSamCoeff1</i>	[1] PET Hargreaves Samani coeff. at level 1
in, out	<i>real(dp), dimension(:, :, :) :: PrieTayAlpha1</i>	[1] PET Priestley Taylor coeff. at level 1
in, out	<i>real(dp), dimension(:, :, :) :: aeroResist1</i>	[s m-1] PET aerodynamical resistance at level 1
in, out	<i>real(dp), dimension(:, :, :) :: surfResist1</i>	[s m-1] PET bulk surface resistance at level 1
in, out	<i>real(dp), dimension(:, :, :) :: fRoots1</i>	fraction of roots in soil horizon
in, out	<i>real(dp), dimension(:, :, :) :: kFastFlow1</i>	[10^-3 m] Recession coefficient of the upper reservoir, upper outlet
in, out	<i>real(dp), dimension(:, :, :) :: kSlowFlow1</i>	[10^-3 m] Recession coefficient of the upper reservoir, lower outlet
in, out	<i>real(dp), dimension(:, :, :) :: kBaseFlow1</i>	Level 1 baseflow recession
in, out	<i>real(dp), dimension(:, :, :) :: kPerco1</i>	[d-1] percolation coefficient
in, out	<i>real(dp), dimension(:, :, :) :: karstLoss1</i>	
in, out	<i>real(dp), dimension(:, :, :) :: soilMoistFC1</i>	[10^-3 m] field capacity
in, out	<i>real(dp), dimension(:, :, :) :: soilMoistSat1</i>	[10^-3 m] depth of saturated SM
in, out	<i>real(dp), dimension(:, :, :) :: soilMoistExp1</i>	Parameter that determines the rel. contribution to SM, upscal. Bulk den.
in, out	<i>real(dp), dimension(:, :, :) :: jarvis_thresh_c1</i>	[1] jarvis critical value for norm SWC
in, out	<i>real(dp), dimension(:, :, :) :: tempThresh1</i>	[degreeC] threshold temperature for snow rain
in, out	<i>real(dp), dimension(:, :, :) :: unsatThresh1</i>	[10^-3 m] Threshhold water depth in upper reservoir (for Runoff contribution)
in, out	<i>real(dp), dimension(:, :, :) :: sealedThresh1</i>	threshold parameter
in, out	<i>real(dp), dimension(:, :, :) :: wiltingPoint1</i>	[10^-3 m] permanent wilting point

Parameters

in, out	<i>real(dp), dimension(:, :, :) :: maxInter1</i>	
in, out	<i>real(dp), dimension(:, :, :) :: petLAICorFactor</i>	
in	<i>real(dp), dimension(:, optional :: parameterset</i>	

Authors

Stephan Thober, Rohini Kumar

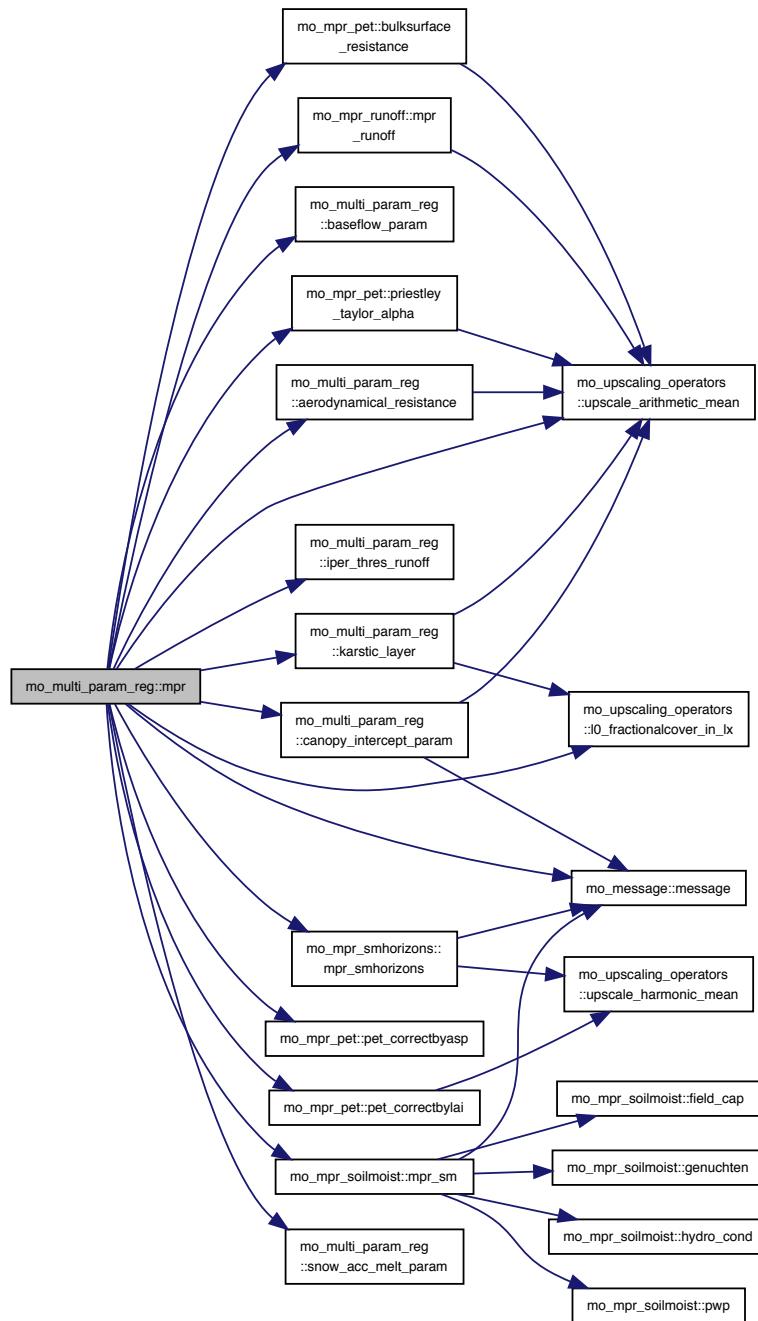
Date

Dec 2012

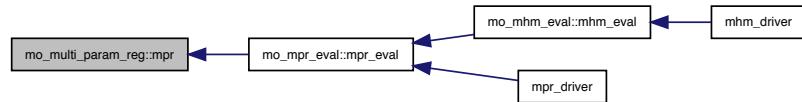
References aerodynamical_resistance(), baseflow_param(), mo_mpr_pet::bulksurface_resistance(), mo_mpr_global_variables::c2stu, canopy_intercept_param(), mo_mpr_global_variables::fracsealed_cityarea, mo_common_variables::global_parameters, mo_mpr_global_variables::horizondepth_mhm, mo_mpr_global_variables::iflag_soildb, iper_thres_runoff(), karstic_layer(), mo_upscaling_operators::l0_fractionalcover_in_lx(), mo_message::message(), mo_mpr_runoff::mpr_runoff(), mo_mpr_soilmoist::mpr_sm(), mo_mpr_smhorizons::mpr_smhorizons(), mo_common_constants::nodata_dp, mo_common_constants::nodata_i4, mo_mpr_global_variables::nsoilhorizons_mhm, mo_mpr_pet::pet_correctbyasp(), mo_mpr_pet::pet_correctbylai(), mo_mpr_pet::priestley_taylor_alpha(), mo_common_variables::processmatrix, snow_acc_melt_param(), mo_mpr_global_variables::soildb, and mo_upscaling_operators::upscale_arithmetic_mean().

Referenced by mo_mpr_eval::mpr_eval().

Here is the call graph for this function:



Here is the caller graph for this function:



15.62.2.7 snow_acc_melt_param()

```

subroutine mo_multi_param_reg::snow_acc_melt_param (
    real(dp), dimension(8), intent(in) param,
    real(dp), intent(in) c2TSTu,
    real(dp), dimension(:, ), intent(in) fForest1,
    real(dp), dimension(:, ), intent(in) fIperm1,
    real(dp), dimension(:, ), intent(in) fPerm1,
    real(dp), dimension(:, ), intent(out) tempThresh1,
    real(dp), dimension(:, ), intent(out) degDayNoPre1,
    real(dp), dimension(:, ), intent(out) degDayIncl1,
    real(dp), dimension(:, ), intent(out) degDayMax1 )
  
```

Calculates the snow parameters.

This subroutine calculates the snow parameters threshold temperature (TT), degree-day factor without precipitation (DD) and maximum degree-day factor (DDmax) as well as increase of degree-day factor per mm of increase in precipitation (IDDP). Global parameters needed (see mhm_parameter.nml):

- param(1) = snowTresholdTemperature
- param(2) = degreeDayFactor_forest
- param(3) = degreeDayFactor_impervious
- param(4) = degreeDayFactor_pervious
- param(5) = increaseDegreeDayFactorByPrecip
- param(6) = maxDegreeDayFactor_forest
- param(7) = maxDegreeDayFactor_impervious
- param(8) = maxDegreeDayFactor_pervious INTENT(IN)

Parameters

in	<i>real(dp), dimension(8) :: param</i>	eight global parameters
in	<i>real(dp) :: c2TSTu</i>	- unit transformation coefficient
in	<i>real(dp), dimension(:,) :: fForest1</i>	[1] fraction of forest cover
in	<i>real(dp), dimension(:,) :: fIperm1</i>	[1] fraction of sealed area
in	<i>real(dp), dimension(:,) :: fPerm1</i>	[1] fraction of permeable area
out	<i>real(dp), dimension(:,) :: tempThresh1</i>	[degreeC] threshold temperature for snow rain
out	<i>real(dp), dimension(:,) :: degDayNoPre1</i>	[mm-1 degreeC-1] Degree-day factor with
out	<i>real(dp), dimension(:,) :: degDayIncl1</i>	[d-1 degreeC-1] Increase of the Degree-day
out	<i>real(dp), dimension(:,) :: degDayMax1</i>	[mm-1 degreeC-1] Maximum Degree-day factor

Authors

Stephan Thober, Rohini Kumar

Date

Dec 2012

Referenced by `mpr()`.

Here is the caller graph for this function:



15.63 mo_ncread Module Reference

Data Types

- interface [get_ncvar](#)

Functions/Subroutines

- integer(i4) function, dimension(5), public [get_ncdim](#) (Filename, Variable, PrintInfo, ndims)
- subroutine, public [get_ncdimatt](#) (Filename, Variable, DimName, DimLen)
- subroutine, public [get_ncvaratt](#) (FileName, VarName, AttName, AttValues, fid, dtype)
- subroutine [get_ncvar_0d_sp](#) (Filename, VarName, Dat, fid)
- subroutine [get_ncvar_0d_dp](#) (Filename, VarName, Dat, fid)
- subroutine [get_ncvar_1d_sp](#) (Filename, VarName, Dat, start, a_count, fid)
- subroutine [get_ncvar_1d_dp](#) (Filename, VarName, Dat, start, a_count, fid)
- subroutine [get_ncvar_2d_sp](#) (Filename, VarName, Dat, start, a_count, fid)
- subroutine [get_ncvar_2d_dp](#) (Filename, VarName, Dat, start, a_count, fid)
- subroutine [get_ncvar_3d_sp](#) (Filename, VarName, Dat, start, a_count, fid)
- subroutine [get_ncvar_3d_dp](#) (Filename, VarName, Dat, start, a_count, fid)
- subroutine [get_ncvar_4d_sp](#) (Filename, VarName, Dat, start, a_count, fid)
- subroutine [get_ncvar_4d_dp](#) (Filename, VarName, Dat, start, a_count, fid)
- subroutine [get_ncvar_5d_sp](#) (Filename, VarName, Dat, start, a_count, fid)
- subroutine [get_ncvar_5d_dp](#) (Filename, VarName, Dat, start, a_count, fid)
- subroutine [get_ncvar_0d_i4](#) (Filename, VarName, Dat, fid)
- subroutine [get_ncvar_1d_i4](#) (Filename, VarName, Dat, start, a_count, fid)
- subroutine [get_ncvar_2d_i4](#) (Filename, VarName, Dat, start, a_count, fid)
- subroutine [get_ncvar_3d_i4](#) (Filename, VarName, Dat, start, a_count, fid)
- subroutine [get_ncvar_4d_i4](#) (Filename, VarName, Dat, start, a_count, fid)
- subroutine [get_ncvar_5d_i4](#) (Filename, VarName, Dat, start, a_count, fid)
- subroutine [get_ncvar_0d_i1](#) (Filename, VarName, Dat, fid)
- subroutine [get_ncvar_1d_i1](#) (Filename, VarName, Dat, start, a_count, fid)
- subroutine [get_ncvar_2d_i1](#) (Filename, VarName, Dat, start, a_count, fid)
- subroutine [get_ncvar_3d_i1](#) (Filename, VarName, Dat, start, a_count, fid)
- subroutine [get_ncvar_4d_i1](#) (Filename, VarName, Dat, start, a_count, fid)
- subroutine [get_ncvar_5d_i1](#) (Filename, VarName, Dat, start, a_count, fid)
- integer(i4) function, public [ncopen](#) (Fname)

- subroutine, public [ncclose](#) (ncid)
- subroutine [get_info](#) (Varname, ncid, varid, xtype, dl, Info, ndims)
- subroutine [check](#) (status)

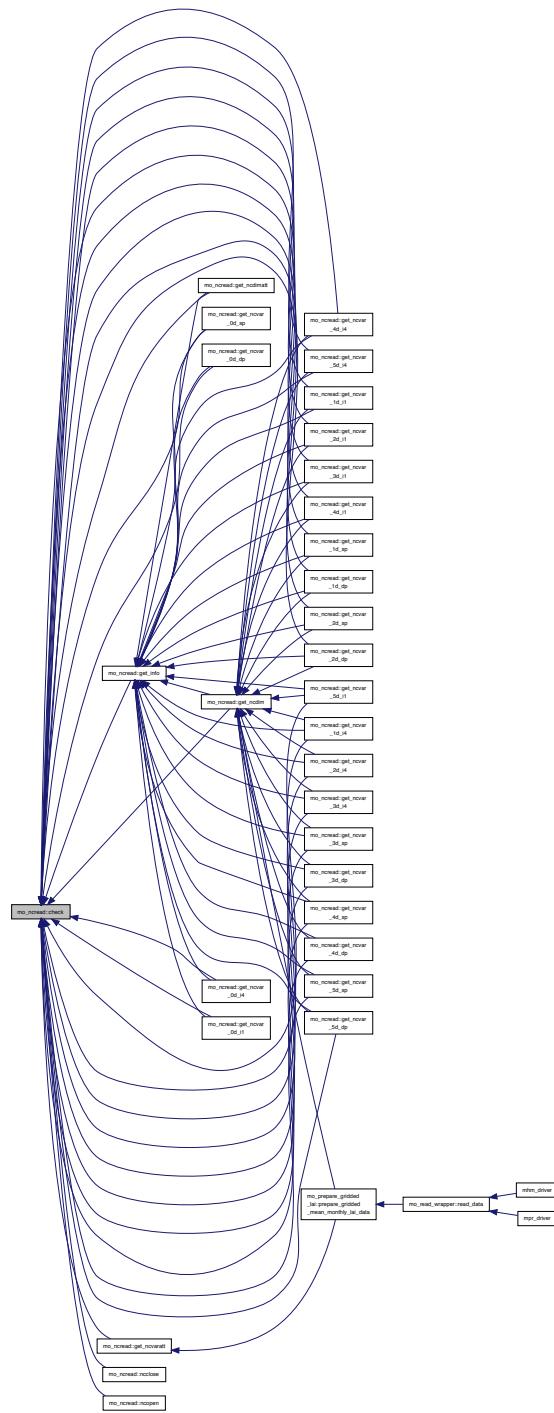
15.63.1 Function/Subroutine Documentation

15.63.1.1 [check\(\)](#)

```
subroutine mo_ncread:::check (
    integer(i4), intent(in) status )  [private]
```

Referenced by [get_info\(\)](#), [get_ncdim\(\)](#), [get_ncdimatt\(\)](#), [get_ncvar_0d_dp\(\)](#), [get_ncvar_0d_i1\(\)](#), [get_ncvar_0d_i4\(\)](#), [get_ncvar_0d_sp\(\)](#), [get_ncvar_1d_dp\(\)](#), [get_ncvar_1d_i1\(\)](#), [get_ncvar_1d_i4\(\)](#), [get_ncvar_1d_sp\(\)](#), [get_ncvar_2d_dp\(\)](#), [get_ncvar_2d_i1\(\)](#), [get_ncvar_2d_i4\(\)](#), [get_ncvar_2d_sp\(\)](#), [get_ncvar_3d_dp\(\)](#), [get_ncvar_3d_i1\(\)](#), [get_ncvar_3d_i4\(\)](#), [get_ncvar_3d_sp\(\)](#), [get_ncvar_4d_dp\(\)](#), [get_ncvar_4d_i1\(\)](#), [get_ncvar_4d_i4\(\)](#), [get_ncvar_4d_sp\(\)](#), [get_ncvar_5d_dp\(\)](#), [get_ncvar_5d_i1\(\)](#), [get_ncvar_5d_i4\(\)](#), [get_ncvar_5d_sp\(\)](#), [get_ncvaratt\(\)](#), [ncclose\(\)](#), and [ncopen\(\)](#).

Here is the caller graph for this function:



15.63.1.2 get_info()

```
subroutine mo_ncread::get_info (
        character(len = *), intent(in) Varname,
        integer(i4), intent(in) ncid,
```

```
integer(i4), intent(out) varid,  
integer(i4), intent(out) xtype,  
integer(i4), dimension(:), intent(inout), optional dl,  
logical, intent(in), optional Info,  
integer(i4), intent(out), optional ndims ) [private]
```

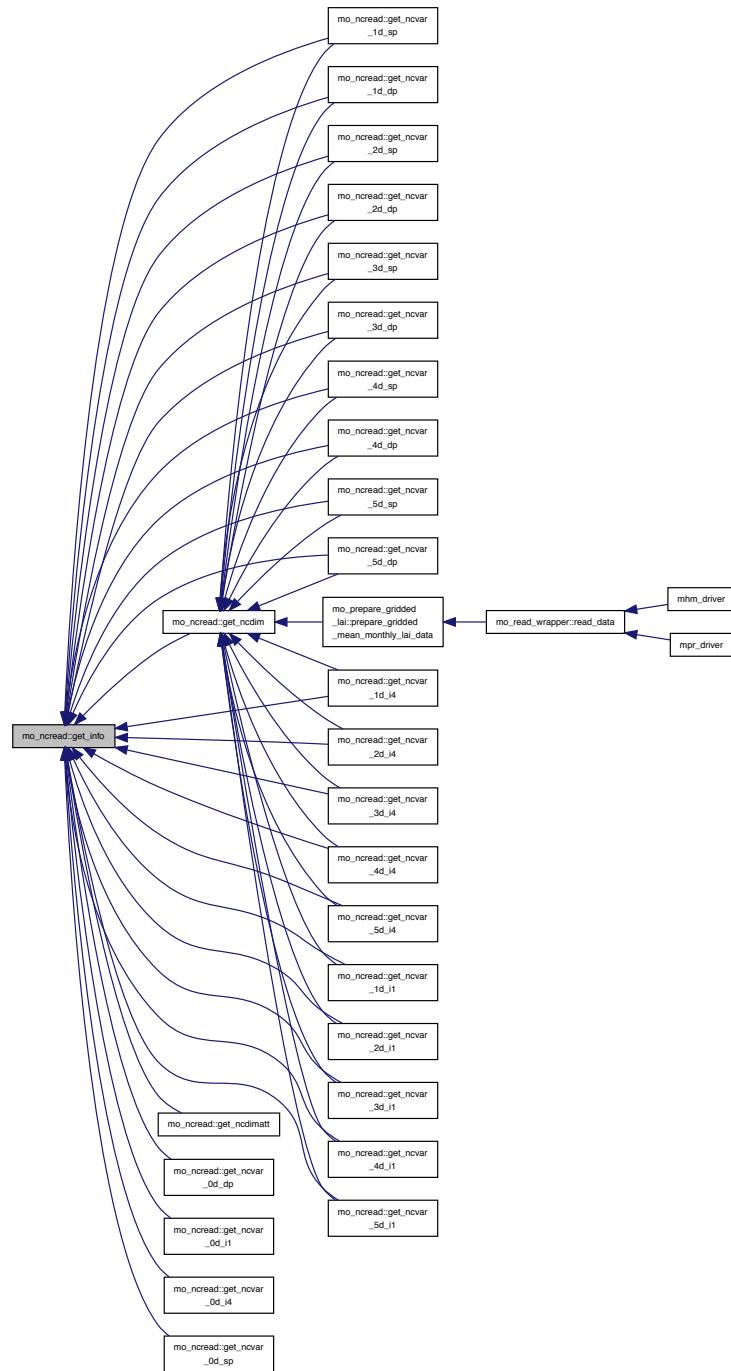
References check().

Referenced by get_ncdim(), get_ncdimatt(), get_ncvar_0d_dp(), get_ncvar_0d_i1(), get_ncvar_0d_i4(), get_ncvar_0d_sp(), get_ncvar_1d_dp(), get_ncvar_1d_i1(), get_ncvar_1d_i4(), get_ncvar_1d_sp(), get_ncvar_2d_dp(), get_ncvar_2d_i1(), get_ncvar_2d_i4(), get_ncvar_2d_sp(), get_ncvar_3d_dp(), get_ncvar_3d_i1(), get_ncvar_3d_i4(), get_ncvar_3d_sp(), get_ncvar_4d_dp(), get_ncvar_4d_i1(), get_ncvar_4d_i4(), get_ncvar_4d_sp(), get_ncvar_5d_dp(), get_ncvar_5d_i1(), get_ncvar_5d_i4(), and get_ncvar_5d_sp().

Here is the call graph for this function:



Here is the caller graph for this function:



15.63.1.3 get_ncdim()

```

integer(i4) function, dimension(5), public mo_ncread::get_ncdim (
    character(len = *), intent(in)  Filename,
    character(len = *), intent(in)  Variable,

```

```
logical, intent(in), optional PrintInfo,  
integer(i4), intent(out), optional ndims )
```

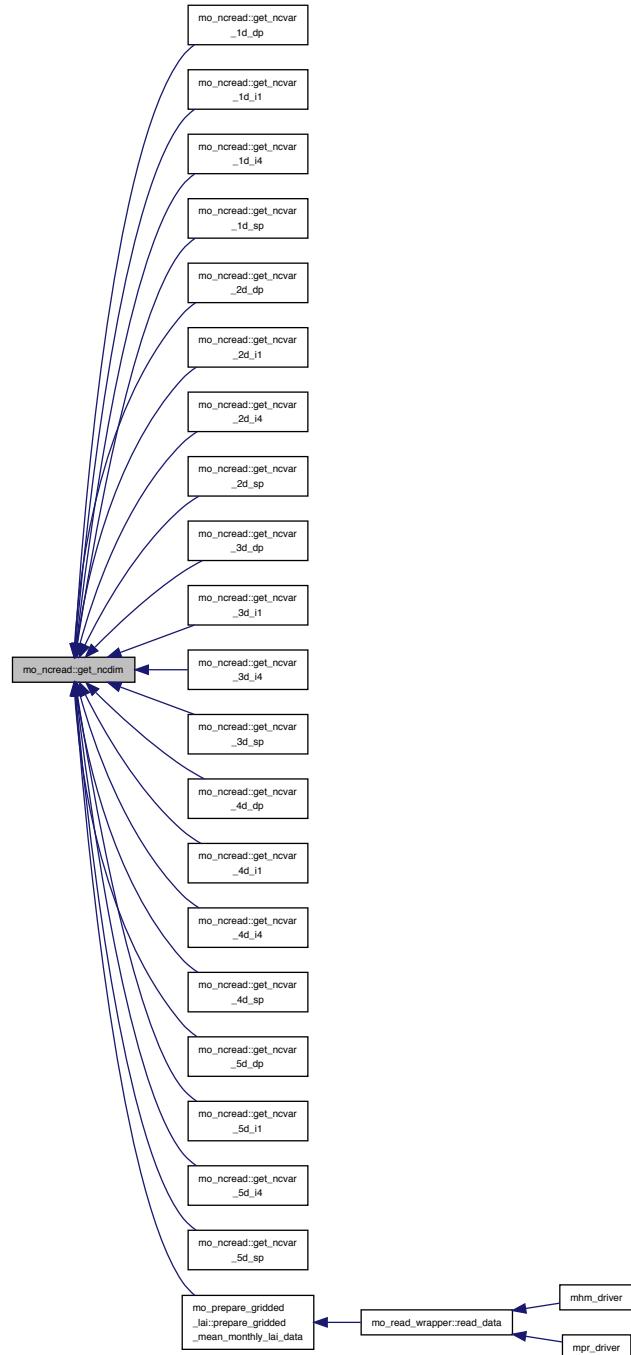
References `check()`, and `get_info()`.

Referenced by `get_ncvar_1d_dp()`, `get_ncvar_1d_i1()`, `get_ncvar_1d_i4()`, `get_ncvar_1d_sp()`, `get_ncvar_2d_dp()`, `get_ncvar_2d_i1()`, `get_ncvar_2d_i4()`, `get_ncvar_2d_sp()`, `get_ncvar_3d_dp()`, `get_ncvar_3d_i1()`, `get_ncvar_3d_i4()`, `get_ncvar_3d_sp()`, `get_ncvar_4d_dp()`, `get_ncvar_4d_i1()`, `get_ncvar_4d_i4()`, `get_ncvar_4d_sp()`, `get_ncvar_5d_dp()`, `get_ncvar_5d_i1()`, `get_ncvar_5d_i4()`, `get_ncvar_5d_sp()`, and `mo_prepare_gridded_lai::prepare_gridded_mean_monthly_lai_data()`.

Here is the call graph for this function:



Here is the caller graph for this function:



15.63.1.4 `get_ncdimatt()`

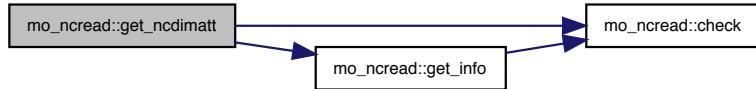
```

subroutine, public mo_ncread::get_ncdimatt (
    character(len = *), intent(in)  Filename,
    character(len = *), intent(in)  Variable,
  
```

```
character(len = *), dimension(:), intent(out), allocatable DimName,
integer(i4), dimension(:), intent(out), optional, allocatable DimLen )
```

References `check()`, and `get_info()`.

Here is the call graph for this function:

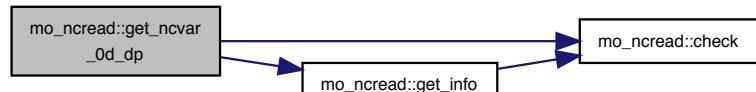


15.63.1.5 `get_ncvar_0d_dp()`

```
subroutine mo_ncread::get_ncvar_0d_dp (
    character(len = *), intent(in) Filename,
    character(len = *), intent(in) VarName,
    real(dp), intent(inout) Dat,
    integer(i4), intent(in), optional fid ) [private]
```

References `check()`, and `get_info()`.

Here is the call graph for this function:

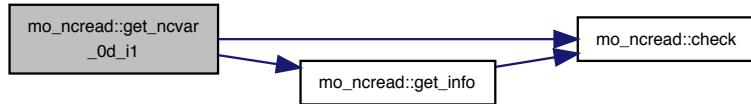


15.63.1.6 `get_ncvar_0d_il()`

```
subroutine mo_ncread::get_ncvar_0d_il (
    character(len = *), intent(in) Filename,
    character(len = *), intent(in) VarName,
    integer(1), intent(inout) Dat,
    integer(i4), intent(in), optional fid ) [private]
```

References `check()`, and `get_info()`.

Here is the call graph for this function:

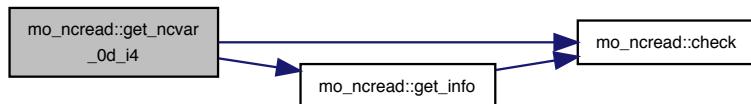


15.63.1.7 get_ncvar_0d_i4()

```
subroutine mo_ncread::get_ncvar_0d_i4 (
    character(len = *), intent(in)  Filename,
    character(len = *), intent(in)  VarName,
    integer(i4), intent(inout)  Dat,
    integer(i4), intent(in), optional fid )  [private]
```

References check(), and get_info().

Here is the call graph for this function:

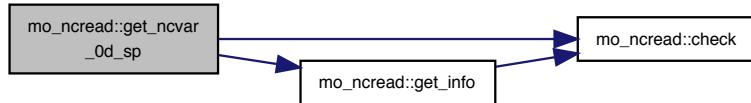


15.63.1.8 get_ncvar_0d_sp()

```
subroutine mo_ncread::get_ncvar_0d_sp (
    character(len = *), intent(in)  Filename,
    character(len = *), intent(in)  VarName,
    real(sp), intent(inout)  Dat,
    integer(i4), intent(in), optional fid )  [private]
```

References check(), and get_info().

Here is the call graph for this function:



15.63.1.9 get_ncvar_1d_dp()

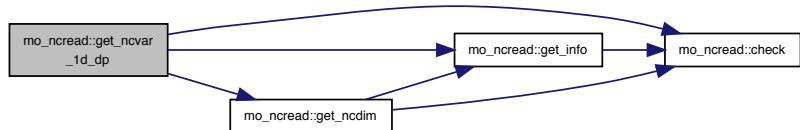
```

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

```

References `check()`, `get_info()`, and `get_ncdim()`.

Here is the call graph for this function:



15.63.1.10 get_ncvar_1d_i1()

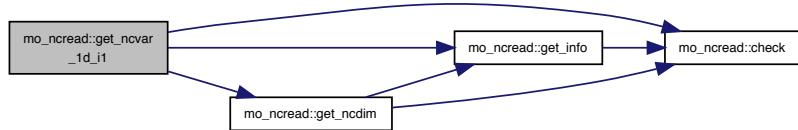
```

subroutine mo_ncread::get_ncvar_1d_i1 (
    character(len = *), intent(in)  Filename,
    character(len = *), intent(in)  VarName,
    integer(l), dimension(:), intent(inout), allocatable Dat,
    integer(i4), dimension(:), intent(in), optional start,
    integer(i4), dimension(:), intent(in), optional a_count,
    integer(i4), intent(in), optional fid )  [private]

```

References `check()`, `get_info()`, and `get_ncdim()`.

Here is the call graph for this function:



15.63.1.11 get_ncvar_1d_i4()

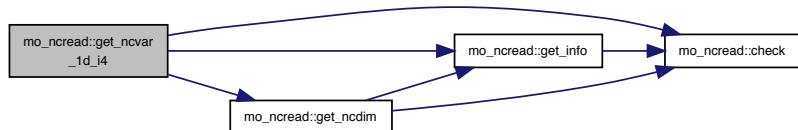
```

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

```

References check(), get_info(), and get_ncdim().

Here is the call graph for this function:



15.63.1.12 get_ncvar_1d_sp()

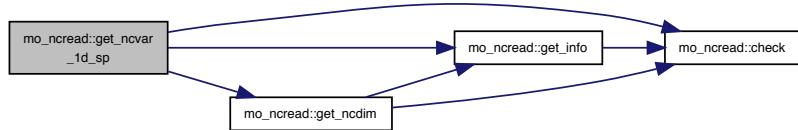
```

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

```

References check(), get_info(), and get_ncdim().

Here is the call graph for this function:



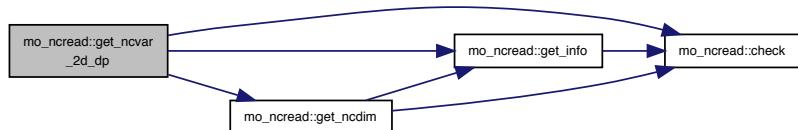
15.63.1.13 get_ncvar_2d_dp()

```

subroutine mo_ncread::get_ncvar_2d_dp (
    character(len = *), intent(in)  Filename,
    character(len = *), intent(in)  VarName,
    real(dp), dimension(:, :), intent(inout), allocatable Dat,
    integer(i4), dimension(:), intent(in), optional start,
    integer(i4), dimension(:), intent(in), optional a_count,
    integer(i4), intent(in), optional fid )  [private]
  
```

References `check()`, `get_info()`, and `get_ncdim()`.

Here is the call graph for this function:



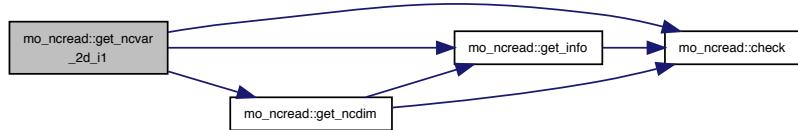
15.63.1.14 get_ncvar_2d_i1()

```

subroutine mo_ncread::get_ncvar_2d_i1 (
    character(len = *), intent(in)  Filename,
    character(len = *), intent(in)  VarName,
    integer(l), dimension(:, :), intent(inout), allocatable Dat,
    integer(i4), dimension(:), intent(in), optional start,
    integer(i4), dimension(:), intent(in), optional a_count,
    integer(i4), intent(in), optional fid )  [private]
  
```

References `check()`, `get_info()`, and `get_ncdim()`.

Here is the call graph for this function:

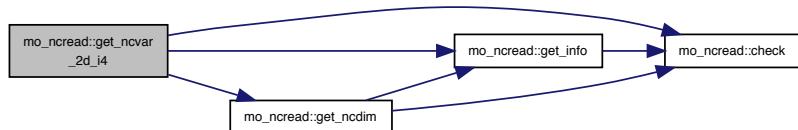


15.63.1.15 get_ncvar_2d_i4()

```
subroutine mo_ncread::get_ncvar_2d_i4 (
    character(len = *), intent(in)  Filename,
    character(len = *), intent(in)  VarName,
    integer(i4), dimension(:, :), intent(inout), allocatable Dat,
    integer(i4), dimension(:), intent(in), optional start,
    integer(i4), dimension(:), intent(in), optional a_count,
    integer(i4), intent(in), optional fid )  [private]
```

References `check()`, `get_info()`, and `get_ncdim()`.

Here is the call graph for this function:

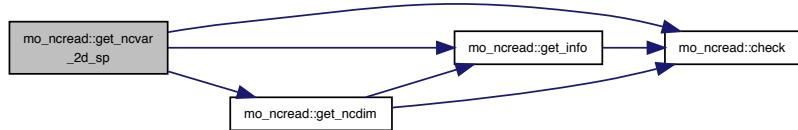


15.63.1.16 get_ncvar_2d_sp()

```
subroutine mo_ncread::get_ncvar_2d_sp (
    character(len = *), intent(in)  Filename,
    character(len = *), intent(in)  VarName,
    real(sp), dimension(:, :), intent(inout), allocatable Dat,
    integer(i4), dimension(:), intent(in), optional start,
    integer(i4), dimension(:), intent(in), optional a_count,
    integer(i4), intent(in), optional fid )  [private]
```

References `check()`, `get_info()`, and `get_ncdim()`.

Here is the call graph for this function:

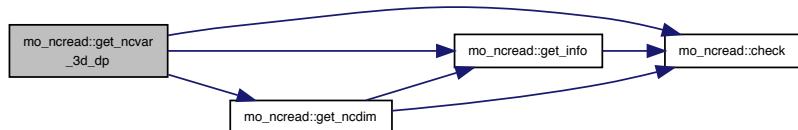


15.63.1.17 get_ncvar_3d_dp()

```
subroutine mo_ncread::get_ncvar_3d_dp (
    character(len = *), intent(in)  Filename,
    character(len = *), intent(in)  VarName,
    real(dp), dimension(:, :, :), intent(inout), allocatable Dat,
    integer(i4), dimension(:), intent(in), optional start,
    integer(i4), dimension(:), intent(in), optional a_count,
    integer(i4), intent(in), optional fid )  [private]
```

References `check()`, `get_info()`, and `get_ncdim()`.

Here is the call graph for this function:

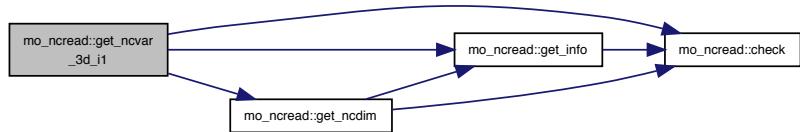


15.63.1.18 get_ncvar_3d_i1()

```
subroutine mo_ncread::get_ncvar_3d_i1 (
    character(len = *), intent(in)  Filename,
    character(len = *), intent(in)  VarName,
    integer(l), dimension(:, :, :), intent(inout), allocatable Dat,
    integer(i4), dimension(:), intent(in), optional start,
    integer(i4), dimension(:), intent(in), optional a_count,
    integer(i4), intent(in), optional fid )  [private]
```

References `check()`, `get_info()`, and `get_ncdim()`.

Here is the call graph for this function:

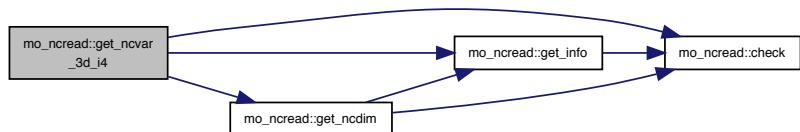


15.63.1.19 get_ncvar_3d_i4()

```
subroutine mo_ncread::get_ncvar_3d_i4 (
    character(len = *), intent(in)  Filename,
    character(len = *), intent(in)  VarName,
    integer(i4), dimension(:, :, :), intent(inout), allocatable  Dat,
    integer(i4), dimension(:), intent(in), optional  start,
    integer(i4), dimension(:), intent(in), optional  a_count,
    integer(i4), intent(in), optional  fid )  [private]
```

References `check()`, `get_info()`, and `get_ncdim()`.

Here is the call graph for this function:

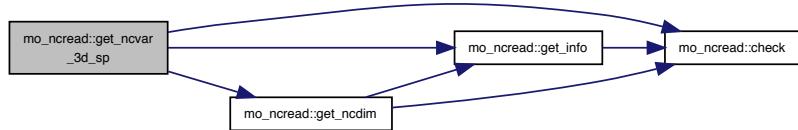


15.63.1.20 get_ncvar_3d_sp()

```
subroutine mo_ncread::get_ncvar_3d_sp (
    character(len = *), intent(in)  Filename,
    character(len = *), intent(in)  VarName,
    real(sp), dimension(:, :, :), intent(inout), allocatable  Dat,
    integer(i4), dimension(:), intent(in), optional  start,
    integer(i4), dimension(:), intent(in), optional  a_count,
    integer(i4), intent(in), optional  fid )  [private]
```

References `check()`, `get_info()`, and `get_ncdim()`.

Here is the call graph for this function:



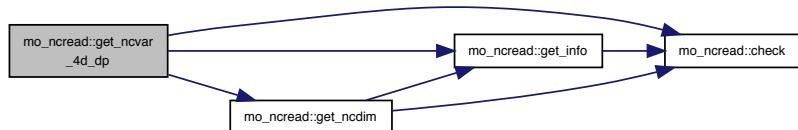
15.63.1.21 get_ncvar_4d_dp()

```

subroutine mo_ncread::get_ncvar_4d_dp (
    character(len = *), intent(in)  Filename,
    character(len = *), intent(in)  VarName,
    real(dp), dimension(:, :, :, :), intent(inout), allocatable Dat,
    integer(i4), dimension(:), intent(in), optional start,
    integer(i4), dimension(:), intent(in), optional a_count,
    integer(i4), intent(in), optional fid )  [private]
  
```

References `check()`, `get_info()`, and `get_ncdim()`.

Here is the call graph for this function:



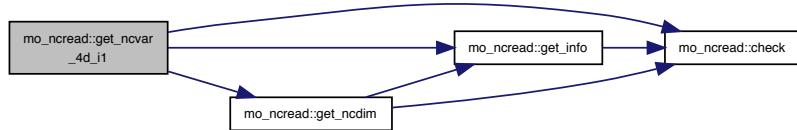
15.63.1.22 get_ncvar_4d_i1()

```

subroutine mo_ncread::get_ncvar_4d_i1 (
    character(len = *), intent(in)  Filename,
    character(len = *), intent(in)  VarName,
    integer(l), dimension(:, :, :, :), intent(inout), allocatable Dat,
    integer(i4), dimension(:), intent(in), optional start,
    integer(i4), dimension(:), intent(in), optional a_count,
    integer(i4), intent(in), optional fid )  [private]
  
```

References `check()`, `get_info()`, and `get_ncdim()`.

Here is the call graph for this function:

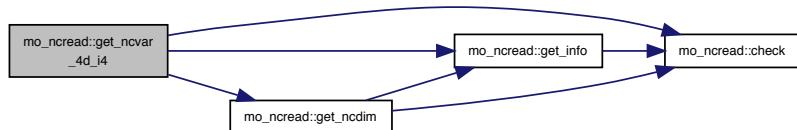


15.63.1.23 get_ncvar_4d_i4()

```
subroutine mo_ncread::get_ncvar_4d_i4 (
    character(len = *), intent(in)  Filename,
    character(len = *), intent(in)  VarName,
    integer(i4), dimension(:, :, :, :), intent(inout), allocatable Dat,
    integer(i4), dimension(:), intent(in), optional start,
    integer(i4), dimension(:), intent(in), optional a_count,
    integer(i4), intent(in), optional fid )  [private]
```

References check(), get_info(), and get_ncdim().

Here is the call graph for this function:

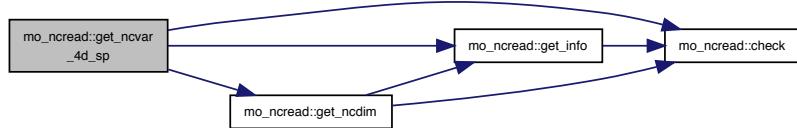


15.63.1.24 get_ncvar_4d_sp()

```
subroutine mo_ncread::get_ncvar_4d_sp (
    character(len = *), intent(in)  Filename,
    character(len = *), intent(in)  VarName,
    real(sp), dimension(:, :, :, :), intent(inout), allocatable Dat,
    integer(i4), dimension(:), intent(in), optional start,
    integer(i4), dimension(:), intent(in), optional a_count,
    integer(i4), intent(in), optional fid )  [private]
```

References check(), get_info(), and get_ncdim().

Here is the call graph for this function:



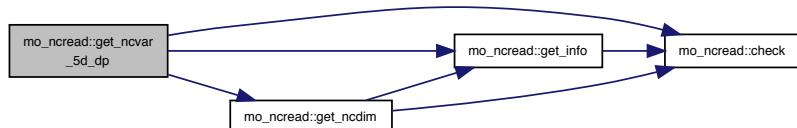
15.63.1.25 `get_ncvar_5d_dp()`

```

subroutine mo_ncread::get_ncvar_5d_dp (
    character(len = *), intent(in)  Filename,
    character(len = *), intent(in)  VarName,
    real(dp), dimension(:, :, :, :, :), intent(inout), allocatable Dat,
    integer(i4), dimension(:), intent(in), optional start,
    integer(i4), dimension(:), intent(in), optional a_count,
    integer(i4), intent(in), optional fid )  [private]
  
```

References `check()`, `get_info()`, and `get_ncdim()`.

Here is the call graph for this function:



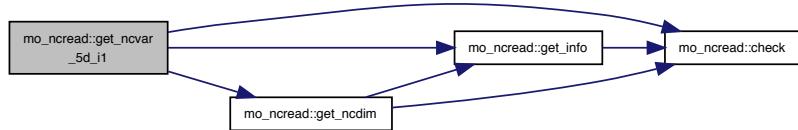
15.63.1.26 `get_ncvar_5d_i1()`

```

subroutine mo_ncread::get_ncvar_5d_i1 (
    character(len = *), intent(in)  Filename,
    character(len = *), intent(in)  VarName,
    integer(l), dimension(:, :, :, :, :), intent(inout), allocatable Dat,
    integer(i4), dimension(:), intent(in), optional start,
    integer(i4), dimension(:), intent(in), optional a_count,
    integer(i4), intent(in), optional fid )  [private]
  
```

References `check()`, `get_info()`, and `get_ncdim()`.

Here is the call graph for this function:



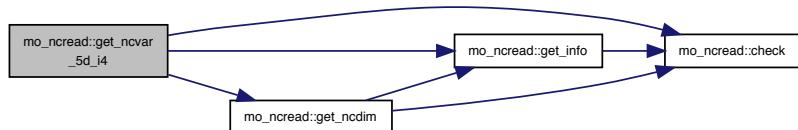
15.63.1.27 get_ncvar_5d_i4()

```

subroutine mo_ncread::get_ncvar_5d_i4 (
    character(len = *), intent(in)  Filename,
    character(len = *), intent(in)  VarName,
    integer(i4), dimension(:, :, :, :, :), intent(inout), allocatable Dat,
    integer(i4), dimension(:), intent(in), optional start,
    integer(i4), dimension(:), intent(in), optional a_count,
    integer(i4), intent(in), optional fid )  [private]
  
```

References check(), get_info(), and get_ncdim().

Here is the call graph for this function:



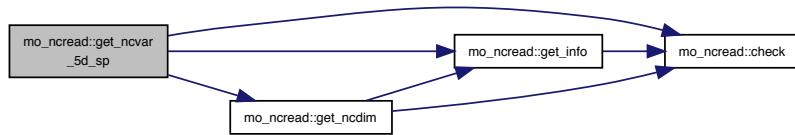
15.63.1.28 get_ncvar_5d_sp()

```

subroutine mo_ncread::get_ncvar_5d_sp (
    character(len = *), intent(in)  Filename,
    character(len = *), intent(in)  VarName,
    real(sp), dimension(:, :, :, :, :), intent(inout), allocatable Dat,
    integer(i4), dimension(:), intent(in), optional start,
    integer(i4), dimension(:), intent(in), optional a_count,
    integer(i4), intent(in), optional fid )  [private]
  
```

References check(), get_info(), and get_ncdim().

Here is the call graph for this function:



15.63.1.29 get_ncvaratt()

```
subroutine, public mo_ncread::get_ncvaratt (
    character(len = *), intent(in) FileName,
    character(len = *), intent(in) VarName,
    character(len = *), intent(in) AttName,
    character(len = *), intent(out) AttValues,
    integer(i4), intent(in), optional fid,
    integer(i4), intent(out), optional dtype )
```

References check().

Referenced by [mo_prepare_gridded_lai::prepare_gridded_mean_monthly_lai_data\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



15.63.1.30 ncclose()

```
subroutine, public mo_ncread::ncclose (
    integer(i4), intent(in) ncid )
```

References check().

Here is the call graph for this function:



15.63.1.31 ncopen()

```
integer(i4) function, public mo_ncread::ncopen (
    character(len = *), intent(in) Fname )
```

References check().

Here is the call graph for this function:



15.64 mo_ncwrite Module Reference

Data Types

- type [attribute](#)
- type [dims](#)
- interface [dump_ncdf](#)
- interface [var2nc](#)
Extended [dump_ncdf](#) for multiple variables.
- type [variable](#)

Functions/Subroutines

- subroutine, public [close_ncdf](#) (ncid)
- subroutine, public [create_ncdf](#) (Filename, ncid, lfs, netcdf4, deflate_level)
- subroutine [dump_ncdf_1d_sp](#) (filename, arr, append, lfs, netcdf4, deflate_level)
- subroutine [dump_ncdf_2d_sp](#) (filename, arr, append, lfs, netcdf4, deflate_level)
- subroutine [dump_ncdf_3d_sp](#) (filename, arr, append, lfs, netcdf4, deflate_level)
- subroutine [dump_ncdf_4d_sp](#) (filename, arr, append, lfs, netcdf4, deflate_level)

- subroutine `dump_netcdf_5d_sp` (filename, arr, append, lfs, netcdf4, deflate_level)
- subroutine `dump_netcdf_1d_dp` (filename, arr, append, lfs, netcdf4, deflate_level)
- subroutine `dump_netcdf_2d_dp` (filename, arr, append, lfs, netcdf4, deflate_level)
- subroutine `dump_netcdf_3d_dp` (filename, arr, append, lfs, netcdf4, deflate_level)
- subroutine `dump_netcdf_4d_dp` (filename, arr, append, lfs, netcdf4, deflate_level)
- subroutine `dump_netcdf_5d_dp` (filename, arr, append, lfs, netcdf4, deflate_level)
- subroutine `dump_netcdf_1d_i4` (filename, arr, append, lfs, netcdf4, deflate_level)
- subroutine `dump_netcdf_2d_i4` (filename, arr, append, lfs, netcdf4, deflate_level)
- subroutine `dump_netcdf_3d_i4` (filename, arr, append, lfs, netcdf4, deflate_level)
- subroutine `dump_netcdf_4d_i4` (filename, arr, append, lfs, netcdf4, deflate_level)
- subroutine `dump_netcdf_5d_i4` (filename, arr, append, lfs, netcdf4, deflate_level)
- subroutine `var2nc_1d_i4` (f_name, arr, dnames, v_name, dim_unlimited, long_name, units, missing_value, attributes, create, ncid, nrec)
- subroutine `var2nc_1d_sp` (f_name, arr, dnames, v_name, dim_unlimited, long_name, units, missing_value, attributes, create, ncid, nrec)
- subroutine `var2nc_1d_dp` (f_name, arr, dnames, v_name, dim_unlimited, long_name, units, missing_value, attributes, create, ncid, nrec)
- subroutine `var2nc_2d_i4` (f_name, arr, dnames, v_name, dim_unlimited, long_name, units, missing_value, attributes, create, ncid, nrec)
- subroutine `var2nc_2d_sp` (f_name, arr, dnames, v_name, dim_unlimited, long_name, units, missing_value, attributes, create, ncid, nrec)
- subroutine `var2nc_2d_dp` (f_name, arr, dnames, v_name, dim_unlimited, long_name, units, missing_value, attributes, create, ncid, nrec)
- subroutine `var2nc_3d_i4` (f_name, arr, dnames, v_name, dim_unlimited, long_name, units, missing_value, attributes, create, ncid, nrec)
- subroutine `var2nc_3d_sp` (f_name, arr, dnames, v_name, dim_unlimited, long_name, units, missing_value, attributes, create, ncid, nrec)
- subroutine `var2nc_3d_dp` (f_name, arr, dnames, v_name, dim_unlimited, long_name, units, missing_value, attributes, create, ncid, nrec)
- subroutine `var2nc_4d_i4` (f_name, arr, dnames, v_name, dim_unlimited, long_name, units, missing_value, attributes, create, ncid, nrec)
- subroutine `var2nc_4d_sp` (f_name, arr, dnames, v_name, dim_unlimited, long_name, units, missing_value, attributes, create, ncid, nrec)
- subroutine `var2nc_4d_dp` (f_name, arr, dnames, v_name, dim_unlimited, long_name, units, missing_value, attributes, create, ncid, nrec)
- subroutine `var2nc_5d_i4` (f_name, arr, dnames, v_name, dim_unlimited, long_name, units, missing_value, attributes, create, ncid, nrec)
- subroutine `var2nc_5d_sp` (f_name, arr, dnames, v_name, dim_unlimited, long_name, units, missing_value, attributes, create, ncid, nrec)
- subroutine `var2nc_5d_dp` (f_name, arr, dnames, v_name, dim_unlimited, long_name, units, missing_value, attributes, create, ncid, nrec)
- subroutine, public `write_dynamic_netcdf` (nclid, irec)
- subroutine, public `write_static_netcdf` (nclid)
- integer(i4) function `open_netcdf` (f_name, create)
- subroutine `check` (status)

Variables

- integer(i4), parameter, public `nmaxdim` = 5
- integer(i4), parameter, public `nmaxatt` = 20
- integer(i4), parameter, public `maxlen` = 256
- integer(i4), parameter, public `ngatt` = 20
- integer(i4), parameter, public `nattdim` = 2
- integer(i4), public `nvargs`
- integer(i4), public `ndims`
- type(`dims`), dimension(:), allocatable, public `dnc`
- type(`variable`), dimension(:), allocatable, public `v`
- type(`attribute`), dimension(`ngatt`), public `gatt`

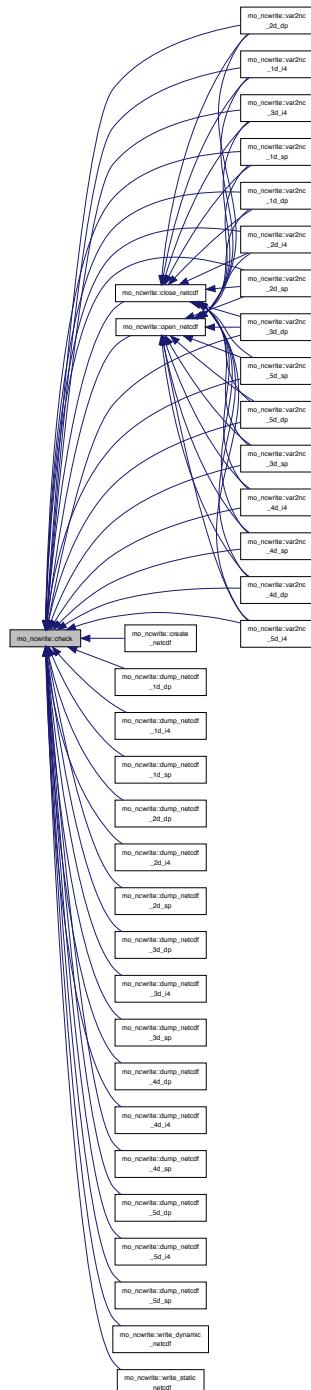
15.64.1 Function/Subroutine Documentation

15.64.1.1 check()

```
subroutine mo_ncwrite::check (
    integer(i4), intent(in) status )  [private]
```

Referenced by `close_netcdf()`, `create_netcdf()`, `dump_netcdf_1d_dp()`, `dump_netcdf_1d_i4()`, `dump_netcdf_1d_sp()`, `dump_netcdf_2d_dp()`, `dump_netcdf_2d_i4()`, `dump_netcdf_2d_sp()`, `dump_netcdf_3d_dp()`, `dump_netcdf_3d_i4()`, `dump_netcdf_3d_sp()`, `dump_netcdf_4d_dp()`, `dump_netcdf_4d_i4()`, `dump_netcdf_4d_sp()`, `dump_netcdf_5d_dp()`, `dump_netcdf_5d_i4()`, `dump_netcdf_5d_sp()`, `open_netcdf()`, `var2nc_1d_dp()`, `var2nc_1d_i4()`, `var2nc_1d_sp()`, `var2nc_2d_dp()`, `var2nc_2d_i4()`, `var2nc_2d_sp()`, `var2nc_3d_dp()`, `var2nc_3d_i4()`, `var2nc_3d_sp()`, `var2nc_4d_dp()`, `var2nc_4d_i4()`, `var2nc_4d_sp()`, `var2nc_5d_dp()`, `var2nc_5d_i4()`, `var2nc_5d_sp()`, `write_dynamic_netcdf()`, and `write_static_netcdf()`.

Here is the caller graph for this function:



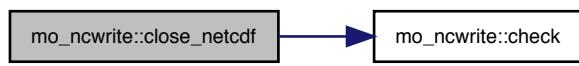
15.64.1.2 `close_netcdf()`

```
subroutine, public mo_ncwrite::close_netcdf (
    integer(i4), intent(in) ncId )
```

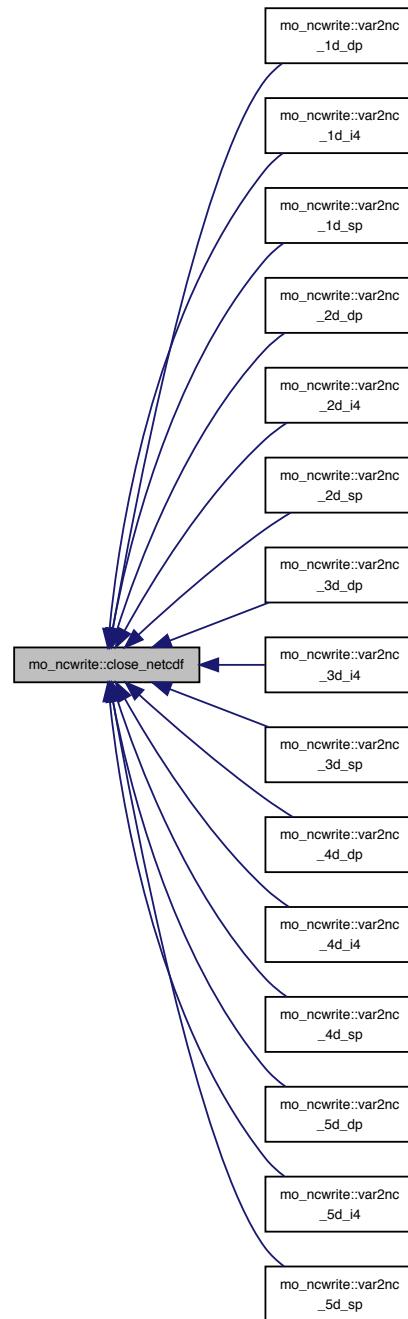
References `check()`.

Referenced by var2nc_1d_dp(), var2nc_1d_i4(), var2nc_1d_sp(), var2nc_2d_dp(), var2nc_2d_i4(), var2nc_2d_sp(), var2nc_3d_dp(), var2nc_3d_i4(), var2nc_3d_sp(), var2nc_4d_dp(), var2nc_4d_i4(), var2nc_4d_sp(), var2nc_5d_← dp(), var2nc_5d_i4(), and var2nc_5d_sp().

Here is the call graph for this function:



Here is the caller graph for this function:



15.64.1.3 create_ncdf()

```
subroutine, public mo_ncwrite::create_ncdf (
    character(len = *), intent(in)  Filename,
    integer(i4), intent(out)  ncid,
```

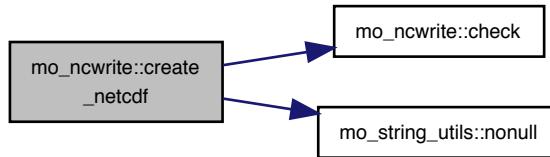
```

logical, intent(in), optional lfs,
logical, intent(in), optional netcdf4,
integer(i4), intent(in), optional deflate_level )

```

References `check()`, `dnc`, `gatt`, `ndims`, `ngatt`, `mo_string_utils::nonnull()`, `nvars`, and `v`.

Here is the call graph for this function:



15.64.1.4 dump_ncdf_1d_dp()

```

subroutine mo_ncwrite::dump_ncdf_1d_dp (
    character(len = *), intent(in) filename,
    real(dp), dimension(:), intent(in) arr,
    logical, intent(in), optional append,
    logical, intent(in), optional lfs,
    logical, intent(in), optional netcdf4,
    integer(i4), intent(in), optional deflate_level ) [private]

```

References `check()`, and `ndims`.

Here is the call graph for this function:



15.64.1.5 dump_ncdf_1d_i4()

```

subroutine mo_ncwrite::dump_ncdf_1d_i4 (
    character(len = *), intent(in) filename,
    integer(i4), dimension(:), intent(in) arr,
    logical, intent(in), optional append,
    logical, intent(in), optional lfs,

```

```
logical, intent(in), optional netcdf4,
integer(i4), intent(in), optional deflate_level ) [private]
```

References check(), and ndims.

Here is the call graph for this function:



15.64.1.6 dump_ncdf_1d_sp()

```
subroutine mo_ncwrite::dump_ncdf_1d_sp (
    character(len = *), intent(in) filename,
    real(sp), dimension(:), intent(in) arr,
    logical, intent(in), optional append,
    logical, intent(in), optional lfs,
    logical, intent(in), optional netcdf4,
    integer(i4), intent(in), optional deflate_level ) [private]
```

References check(), and ndims.

Here is the call graph for this function:



15.64.1.7 dump_ncdf_2d_dp()

```
subroutine mo_ncwrite::dump_ncdf_2d_dp (
    character(len = *), intent(in) filename,
    real(dp), dimension(:, :), intent(in) arr,
    logical, intent(in), optional append,
    logical, intent(in), optional lfs,
    logical, intent(in), optional netcdf4,
    integer(i4), intent(in), optional deflate_level ) [private]
```

References check(), and ndims.

Here is the call graph for this function:



15.64.1.8 dump_ncdf_2d_i4()

```
subroutine mo_ncwrite::dump_ncdf_2d_i4 (
    character(len = *), intent(in) filename,
    integer(i4), dimension(:, :), intent(in) arr,
    logical, intent(in), optional append,
    logical, intent(in), optional lfs,
    logical, intent(in), optional netcdf4,
    integer(i4), intent(in), optional deflate_level ) [private]
```

References check(), and ndims.

Here is the call graph for this function:



15.64.1.9 dump_ncdf_2d_sp()

```
subroutine mo_ncwrite::dump_ncdf_2d_sp (
    character(len = *), intent(in) filename,
    real(sp), dimension(:, :), intent(in) arr,
    logical, intent(in), optional append,
    logical, intent(in), optional lfs,
    logical, intent(in), optional netcdf4,
    integer(i4), intent(in), optional deflate_level ) [private]
```

References check(), and ndims.

Here is the call graph for this function:



15.64.1.10 dump_ncdf_3d_dp()

```

subroutine mo_ncwrite::dump_ncdf_3d_dp (
    character(len = *), intent(in) filename,
    real(dp), dimension(:, :, :), intent(in) arr,
    logical, intent(in), optional append,
    logical, intent(in), optional lfs,
    logical, intent(in), optional netcdf4,
    integer(i4), intent(in), optional deflate_level ) [private]

```

References check(), and ndims.

Here is the call graph for this function:



15.64.1.11 dump_ncdf_3d_i4()

```

subroutine mo_ncwrite::dump_ncdf_3d_i4 (
    character(len = *), intent(in) filename,
    integer(i4), dimension(:, :, :), intent(in) arr,
    logical, intent(in), optional append,
    logical, intent(in), optional lfs,
    logical, intent(in), optional netcdf4,
    integer(i4), intent(in), optional deflate_level ) [private]

```

References check(), and ndims.

Here is the call graph for this function:



15.64.1.12 dump_ncdf_3d_sp()

```
subroutine mo_ncwrite::dump_ncdf_3d_sp (
    character(len = *), intent(in) filename,
    real(sp), dimension(:, :, :), intent(in) arr,
    logical, intent(in), optional append,
    logical, intent(in), optional lfs,
    logical, intent(in), optional netcdf4,
    integer(i4), intent(in), optional deflate_level ) [private]
```

References check(), and ndims.

Here is the call graph for this function:



15.64.1.13 dump_ncdf_4d_dp()

```
subroutine mo_ncwrite::dump_ncdf_4d_dp (
    character(len = *), intent(in) filename,
    real(dp), dimension(:, :, :, :), intent(in) arr,
    logical, intent(in), optional append,
    logical, intent(in), optional lfs,
    logical, intent(in), optional netcdf4,
    integer(i4), intent(in), optional deflate_level ) [private]
```

References check(), and ndims.

Here is the call graph for this function:



15.64.1.14 dump_ncdf_4d_i4()

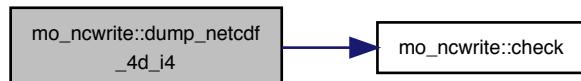
```

subroutine mo_ncwrite::dump_ncdf_4d_i4 (
    character(len = *), intent(in) filename,
    integer(i4), dimension(:, :, :, :), intent(in) arr,
    logical, intent(in), optional append,
    logical, intent(in), optional lfs,
    logical, intent(in), optional netcdf4,
    integer(i4), intent(in), optional deflate_level ) [private]

```

References check(), and ndims.

Here is the call graph for this function:



15.64.1.15 dump_ncdf_4d_sp()

```

subroutine mo_ncwrite::dump_ncdf_4d_sp (
    character(len = *), intent(in) filename,
    real(sp), dimension(:, :, :, :), intent(in) arr,
    logical, intent(in), optional append,
    logical, intent(in), optional lfs,
    logical, intent(in), optional netcdf4,
    integer(i4), intent(in), optional deflate_level ) [private]

```

References check(), and ndims.

Here is the call graph for this function:



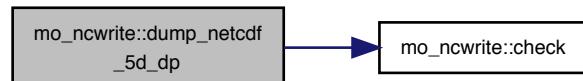
15.64.1.16 dump_ncdf_5d_dp()

```

subroutine mo_ncwrite::dump_ncdf_5d_dp (
    character(len = *), intent(in) filename,
    real(dp), dimension(:, :, :, :, :), intent(in) arr,
    logical, intent(in), optional append,
    logical, intent(in), optional lfs,
    logical, intent(in), optional netcdf4,
    integer(i4), intent(in), optional deflate_level ) [private]
  
```

References check(), and ndims.

Here is the call graph for this function:



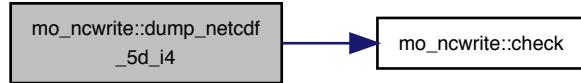
15.64.1.17 dump_ncdf_5d_i4()

```

subroutine mo_ncwrite::dump_ncdf_5d_i4 (
    character(len = *), intent(in) filename,
    integer(i4), dimension(:, :, :, :, :), intent(in) arr,
    logical, intent(in), optional append,
    logical, intent(in), optional lfs,
    logical, intent(in), optional netcdf4,
    integer(i4), intent(in), optional deflate_level ) [private]
  
```

References check(), and ndims.

Here is the call graph for this function:



15.64.1.18 dump_ncdf_5d_sp()

```

subroutine mo_ncwrite::dump_ncdf_5d_sp (
    character(len = *), intent(in) filename,
    real(sp), dimension(:, :, :, :, :), intent(in) arr,
    logical, intent(in), optional append,
    logical, intent(in), optional lfs,
    logical, intent(in), optional netcdf4,
    integer(i4), intent(in), optional deflate_level ) [private]

```

References check(), and ndims.

Here is the call graph for this function:



15.64.1.19 open_ncdf()

```

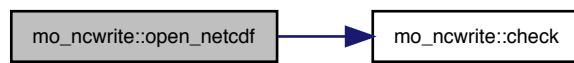
integer(i4) function mo_ncwrite::open_ncdf (
    character(len = *), intent(in) f_name,
    logical, intent(in) create ) [private]

```

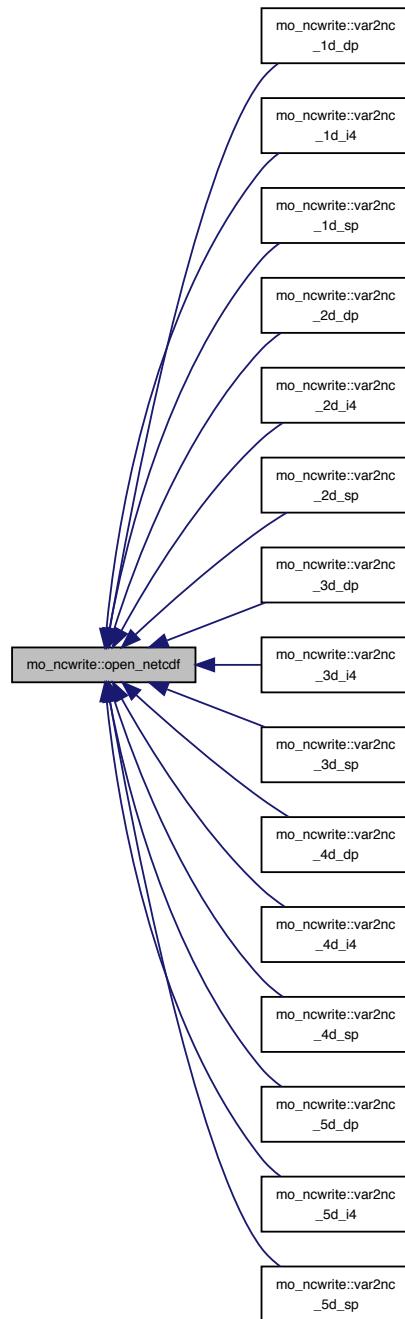
References check().

Referenced by var2nc_1d_dp(), var2nc_1d_i4(), var2nc_1d_sp(), var2nc_2d_dp(), var2nc_2d_i4(), var2nc_2d_sp(), var2nc_3d_dp(), var2nc_3d_i4(), var2nc_3d_sp(), var2nc_4d_dp(), var2nc_4d_i4(), var2nc_4d_sp(), var2nc_5d_dp(), var2nc_5d_i4(), and var2nc_5d_sp().

Here is the call graph for this function:



Here is the caller graph for this function:



15.64.1.20 var2nc_1d_dp()

```
subroutine mo_ncwrite::var2nc_1d_dp (  
    character(len = *), intent(in)  f_name,  
    real(dp), dimension(:), intent(in) arr,
```

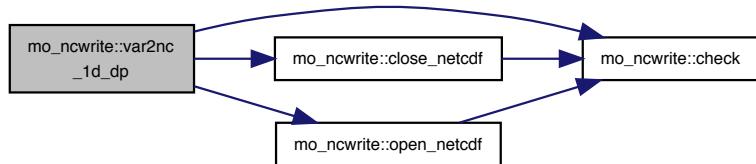
```

character(len = *), dimension(:), intent(in) dnames,
character(len = *), intent(in) v_name,
integer(i4), intent(in), optional dim_unlimited,
character(len = *), intent(in), optional long_name,
character(len = *), intent(in), optional units,
real(dp), intent(in), optional missing_value,
character(256), dimension(:, :), intent(in), optional attributes,
logical, intent(in), optional create,
integer(i4), intent(inout), optional ncid,
integer(i4), intent(in), optional nrec ) [private]

```

References check(), close_ncdf(), ndims, and open_ncdf().

Here is the call graph for this function:



15.64.1.21 var2nc_1d_i4()

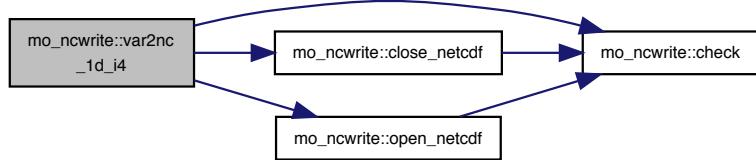
```

subroutine mo_ncwrite::var2nc_1d_i4 (
    character(len = *), intent(in) f_name,
    integer(i4), dimension(:), intent(in) arr,
    character(len = *), dimension(:), intent(in) dnames,
    character(len = *), intent(in) v_name,
    integer(i4), intent(in), optional dim_unlimited,
    character(len = *), intent(in), optional long_name,
    character(len = *), intent(in), optional units,
    integer(i4), intent(in), optional missing_value,
    character(256), dimension(:, :), intent(in), optional attributes,
    logical, intent(in), optional create,
    integer(i4), intent(inout), optional ncid,
    integer(i4), intent(in), optional nrec ) [private]

```

References check(), close_ncdf(), ndims, and open_ncdf().

Here is the call graph for this function:



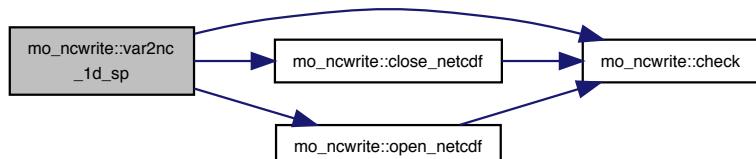
15.64.1.22 var2nc_1d_sp()

```

subroutine mo_ncwrite::var2nc_1d_sp (
    character(len = *), intent(in)  f_name,
    real(sp), dimension(:), intent(in) arr,
    character(len = *), dimension(:), intent(in) dnames,
    character(len = *), intent(in) v_name,
    integer(i4), intent(in), optional dim_unlimited,
    character(len = *), intent(in), optional long_name,
    character(len = *), intent(in), optional units,
    real(sp), intent(in), optional missing_value,
    character(256), dimension(:, :), intent(in), optional attributes,
    logical, intent(in), optional create,
    integer(i4), intent(inout), optional ncid,
    integer(i4), intent(in), optional nrec ) [private]
  
```

References check(), close_ncdf(), ndims, and open_ncdf().

Here is the call graph for this function:



15.64.1.23 var2nc_2d_dp()

```

subroutine mo_ncwrite::var2nc_2d_dp (
    character(len = *), intent(in)  f_name,
    real(dp), dimension(:, :), intent(in) arr,
    character(len = *), dimension(:, ), intent(in) dnames,
  
```

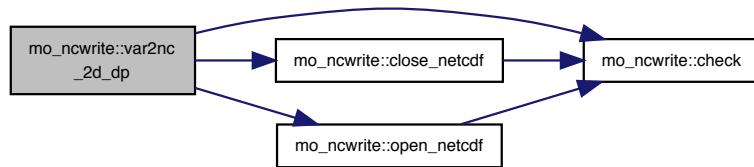
```

character(len = *), intent(in) v_name,
integer(i4), intent(in), optional dim_unlimited,
character(len = *), intent(in), optional long_name,
character(len = *), intent(in), optional units,
real(dp), intent(in), optional missing_value,
character(256), dimension(:, :), intent(in), optional attributes,
logical, intent(in), optional create,
integer(i4), intent(inout), optional ncid,
integer(i4), intent(in), optional nrec ) [private]

```

References `check()`, `close_ncdf()`, `ndims`, and `open_ncdf()`.

Here is the call graph for this function:



15.64.1.24 var2nc_2d_i4()

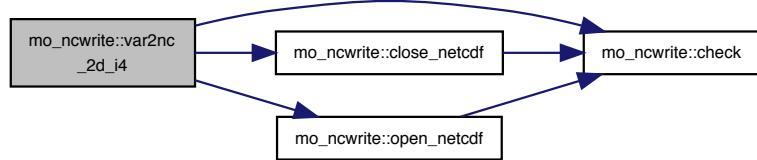
```

subroutine mo_ncwrite::var2nc_2d_i4 (
    character(len = *), intent(in) f_name,
    integer(i4), dimension(:, :), intent(in) arr,
    character(len = *), dimension(:, :), intent(in) dnames,
    character(len = *), intent(in) v_name,
    integer(i4), intent(in), optional dim_unlimited,
    character(len = *), intent(in), optional long_name,
    character(len = *), intent(in), optional units,
    integer(i4), intent(in), optional missing_value,
    character(256), dimension(:, :), intent(in), optional attributes,
    logical, intent(in), optional create,
    integer(i4), intent(inout), optional ncid,
    integer(i4), intent(in), optional nrec ) [private]

```

References `check()`, `close_ncdf()`, `ndims`, and `open_ncdf()`.

Here is the call graph for this function:



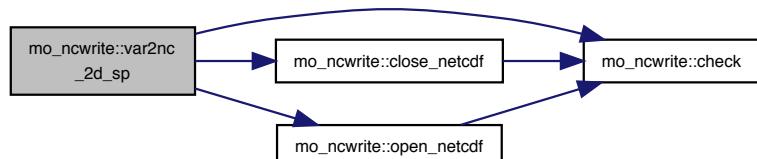
15.64.1.25 var2nc_2d_sp()

```

subroutine mo_ncwrite::var2nc_2d_sp (
    character(len = *), intent(in)  f_name,
    real(sp), dimension(:, :, ), intent(in) arr,
    character(len = *), dimension(:, ), intent(in) dnames,
    character(len = *), intent(in) v_name,
    integer(i4), intent(in), optional dim_unlimited,
    character(len = *), intent(in), optional long_name,
    character(len = *), intent(in), optional units,
    real(sp), intent(in), optional missing_value,
    character(256), dimension(:, :, ), intent(in), optional attributes,
    logical, intent(in), optional create,
    integer(i4), intent(inout), optional ncid,
    integer(i4), intent(in), optional nrec )  [private]
  
```

References check(), close_ncdf(), ndims, and open_ncdf().

Here is the call graph for this function:



15.64.1.26 var2nc_3d_dp()

```

subroutine mo_ncwrite::var2nc_3d_dp (
    character(len = *), intent(in)  f_name,
    real(dp), dimension(:, :, :, ), intent(in) arr,
    character(len = *), dimension(:, ), intent(in) dnames,
  
```

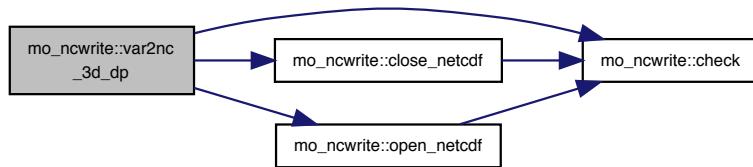
```

character(len = *), intent(in) v_name,
integer(i4), intent(in), optional dim_unlimited,
character(len = *), intent(in), optional long_name,
character(len = *), intent(in), optional units,
real(dp), intent(in), optional missing_value,
character(256), dimension(:, :, ), intent(in), optional attributes,
logical, intent(in), optional create,
integer(i4), intent(inout), optional ncid,
integer(i4), intent(in), optional nrec ) [private]

```

References check(), close_ncdf(), ndims, and open_ncdf().

Here is the call graph for this function:



15.64.1.27 var2nc_3d_i4()

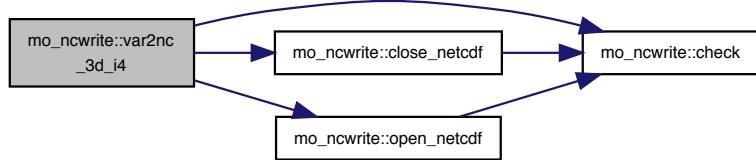
```

subroutine mo_ncwrite::var2nc_3d_i4 (
    character(len = *), intent(in) f_name,
    integer(i4), dimension(:, :, :), intent(in) arr,
    character(len = *), dimension(:, ), intent(in) dnames,
    character(len = *), intent(in) v_name,
    integer(i4), intent(in), optional dim_unlimited,
    character(len = *), intent(in), optional long_name,
    character(len = *), intent(in), optional units,
    integer(i4), intent(in), optional missing_value,
    character(256), dimension(:, :, ), intent(in), optional attributes,
    logical, intent(in), optional create,
    integer(i4), intent(inout), optional ncid,
    integer(i4), intent(in), optional nrec ) [private]

```

References check(), close_ncdf(), ndims, and open_ncdf().

Here is the call graph for this function:



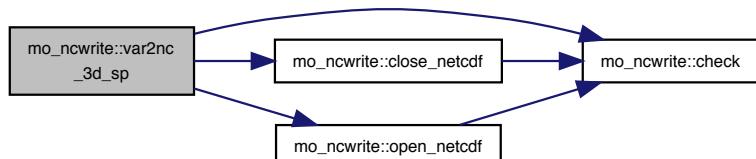
15.64.1.28 var2nc_3d_sp()

```

subroutine mo_ncwrite::var2nc_3d_sp (
    character(len = *), intent(in)  f_name,
    real(sp), dimension(:, :, :), intent(in) arr,
    character(len = *), dimension(:), intent(in) dnames,
    character(len = *), intent(in) v_name,
    integer(i4), intent(in), optional dim_unlimited,
    character(len = *), intent(in), optional long_name,
    character(len = *), intent(in), optional units,
    real(sp), intent(in), optional missing_value,
    character(256), dimension(:, :, :), intent(in), optional attributes,
    logical, intent(in), optional create,
    integer(i4), intent(inout), optional ncid,
    integer(i4), intent(in), optional nrec ) [private]
  
```

References check(), close_ncdf(), ndims, and open_ncdf().

Here is the call graph for this function:



15.64.1.29 var2nc_4d_dp()

```

subroutine mo_ncwrite::var2nc_4d_dp (
    character(len = *), intent(in)  f_name,
    real(dp), dimension(:, :, :, :), intent(in) arr,
    character(len = *), dimension(:), intent(in) dnames,
  
```

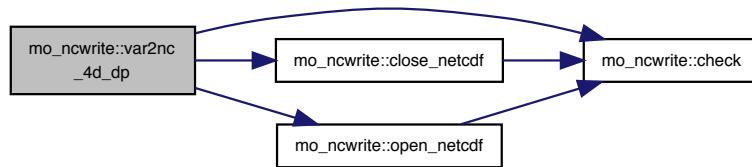
```

character(len = *), intent(in) v_name,
integer(i4), intent(in), optional dim_unlimited,
character(len = *), intent(in), optional long_name,
character(len = *), intent(in), optional units,
real(dp), intent(in), optional missing_value,
character(256), dimension(:, :, :), intent(in), optional attributes,
logical, intent(in), optional create,
integer(i4), intent(inout), optional ncid,
integer(i4), intent(in), optional nrec ) [private]

```

References check(), close_ncdf(), ndims, and open_ncdf().

Here is the call graph for this function:



15.64.1.30 var2nc_4d_i4()

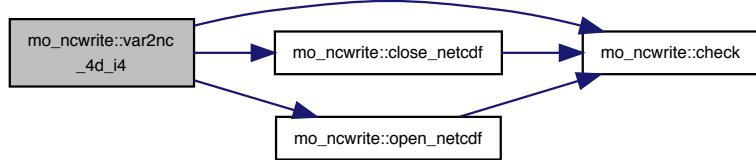
```

subroutine mo_ncwrite::var2nc_4d_i4 (
    character(len = *), intent(in) f_name,
    integer(i4), dimension(:, :, :, :), intent(in) arr,
    character(len = *), dimension(:, :), intent(in) dnames,
    character(len = *), intent(in) v_name,
    integer(i4), intent(in), optional dim_unlimited,
    character(len = *), intent(in), optional long_name,
    character(len = *), intent(in), optional units,
    integer(i4), intent(in), optional missing_value,
    character(256), dimension(:, :, :), intent(in), optional attributes,
    logical, intent(in), optional create,
    integer(i4), intent(inout), optional ncid,
    integer(i4), intent(in), optional nrec ) [private]

```

References check(), close_ncdf(), ndims, and open_ncdf().

Here is the call graph for this function:



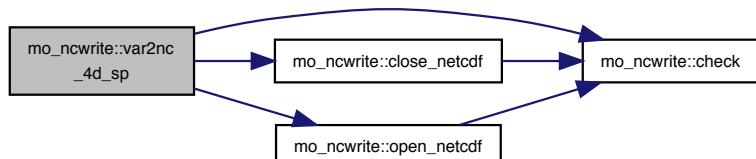
15.64.1.31 var2nc_4d_sp()

```

subroutine mo_ncwrite::var2nc_4d_sp (
    character(len = *), intent(in)  f_name,
    real(sp), dimension(:, :, :, :, :), intent(in) arr,
    character(len = *), dimension(:), intent(in) dnames,
    character(len = *), intent(in) v_name,
    integer(i4), intent(in), optional dim_unlimited,
    character(len = *), intent(in), optional long_name,
    character(len = *), intent(in), optional units,
    real(sp), intent(in), optional missing_value,
    character(256), dimension(:, :, :), intent(in), optional attributes,
    logical, intent(in), optional create,
    integer(i4), intent(inout), optional ncid,
    integer(i4), intent(in), optional nrec ) [private]
  
```

References check(), close_ncdf(), ndims, and open_ncdf().

Here is the call graph for this function:



15.64.1.32 var2nc_5d_dp()

```

subroutine mo_ncwrite::var2nc_5d_dp (
    character(len = *), intent(in)  f_name,
    real(dp), dimension(:, :, :, :, :, :), intent(in) arr,
    character(len = *), dimension(:), intent(in) dnames,
  
```

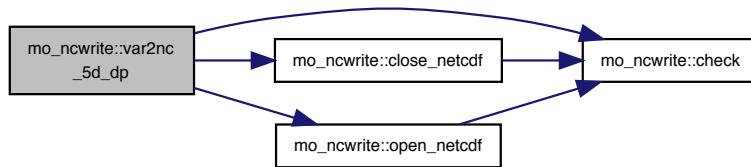
```

character(len = *), intent(in) v_name,
integer(i4), intent(in), optional dim_unlimited,
character(len = *), intent(in), optional long_name,
character(len = *), intent(in), optional units,
real(dp), intent(in), optional missing_value,
character(256), dimension(:, :, :), intent(in), optional attributes,
logical, intent(in), optional create,
integer(i4), intent(inout), optional ncid,
integer(i4), intent(in), optional nrec ) [private]

```

References check(), close_ncdf(), ndims, and open_ncdf().

Here is the call graph for this function:



15.64.1.33 var2nc_5d_i4()

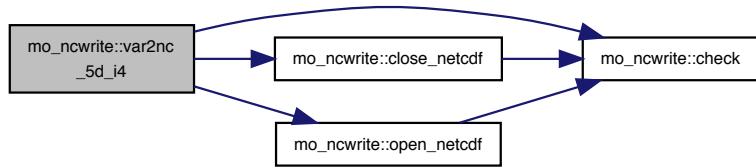
```

subroutine mo_ncwrite::var2nc_5d_i4 (
    character(len = *), intent(in) f_name,
    integer(i4), dimension(:, :, :, :, :), intent(in) arr,
    character(len = *), dimension(:, :), intent(in) dnames,
    character(len = *), intent(in) v_name,
    integer(i4), intent(in), optional dim_unlimited,
    character(len = *), intent(in), optional long_name,
    character(len = *), intent(in), optional units,
    integer(i4), intent(in), optional missing_value,
    character(256), dimension(:, :, :), intent(in), optional attributes,
    logical, intent(in), optional create,
    integer(i4), intent(inout), optional ncid,
    integer(i4), intent(in), optional nrec ) [private]

```

References check(), close_ncdf(), ndims, and open_ncdf().

Here is the call graph for this function:



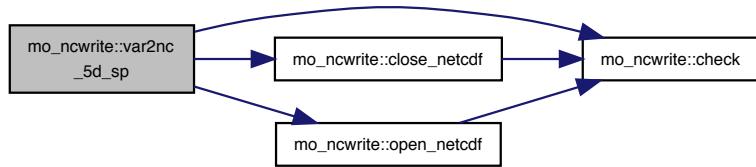
15.64.1.34 var2nc_5d_sp()

```

subroutine mo_ncwrite::var2nc_5d_sp (
    character(len = *), intent(in) f_name,
    real(sp), dimension(:, :, :, :, :), intent(in) arr,
    character(len = *), dimension(:), intent(in) dnames,
    character(len = *), intent(in) v_name,
    integer(i4), intent(in), optional dim_unlimited,
    character(len = *), intent(in), optional long_name,
    character(len = *), intent(in), optional units,
    real(sp), intent(in), optional missing_value,
    character(256), dimension(:, :), intent(in), optional attributes,
    logical, intent(in), optional create,
    integer(i4), intent(inout), optional ncid,
    integer(i4), intent(in), optional nrec ) [private]
  
```

References check(), close_ncdf(), ndims, and open_ncdf().

Here is the call graph for this function:



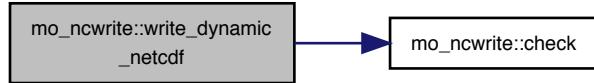
15.64.1.35 write_dynamic_ncdf()

```

subroutine, public mo_ncwrite::write_dynamic_ncdf (
    integer(i4), intent(in) ncId,
    integer(i4), intent(in) irec )
  
```

References check(), nvars, and v.

Here is the call graph for this function:

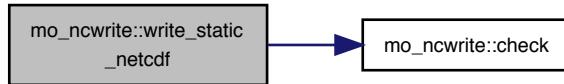


15.64.1.36 write_static_ncdf()

```
subroutine, public mo_ncwrite::write_static_ncdf (
    integer(i4), intent(in) ncId )
```

References check(), nvars, and v.

Here is the call graph for this function:



15.64.2 Variable Documentation

15.64.2.1 dnc

```
type (dims), dimension(:), allocatable, public mo_ncwrite::dnc
```

Referenced by create_ncdf(), and mo_set_ncdf_outputs::set_ncdf().

15.64.2.2 gatt

```
type(attribute), dimension(ngatt), public mo_ncwrite::gatt
```

Referenced by create_ncdf().

15.64.2.3 maxlen

```
integer(i4), parameter, public mo_ncwrite::maxlen = 256
```

15.64.2.4 nattdim

```
integer(i4), parameter, public mo_ncwrite::nattdim = 2
```

15.64.2.5 ndims

```
integer(i4), public mo_ncwrite::ndims
```

Referenced by `create_netcdf()`, `dump_netcdf_1d_dp()`, `dump_netcdf_1d_i4()`, `dump_netcdf_1d_sp()`, `dump_netcdf_2d_dp()`, `dump_netcdf_2d_i4()`, `dump_netcdf_2d_sp()`, `dump_netcdf_3d_dp()`, `dump_netcdf_3d_i4()`, `dump_netcdf_3d_sp()`, `dump_netcdf_4d_dp()`, `dump_netcdf_4d_i4()`, `dump_netcdf_4d_sp()`, `dump_netcdf_5d_dp()`, `dump_netcdf_5d_i4()`, `dump_netcdf_5d_sp()`, `mo_set_netcdf_outputs::set_netcdf()`, `var2nc_1d_dp()`, `var2nc_1d_i4()`, `var2nc_1d_sp()`, `var2nc_2d_dp()`, `var2nc_2d_i4()`, `var2nc_2d_sp()`, `var2nc_3d_dp()`, `var2nc_3d_i4()`, `var2nc_3d_sp()`, `var2nc_4d_dp()`, `var2nc_4d_i4()`, `var2nc_4d_sp()`, `var2nc_5d_dp()`, `var2nc_5d_i4()`, and `var2nc_5d_sp()`.

15.64.2.6 ngatt

```
integer(i4), parameter, public mo_ncwrite::ngatt = 20
```

Referenced by `create_netcdf()`.

15.64.2.7 nmaxatt

```
integer(i4), parameter, public mo_ncwrite::nmaxatt = 20
```

15.64.2.8 nmaxdim

```
integer(i4), parameter, public mo_ncwrite::nmaxdim = 5
```

15.64.2.9 nvars

```
integer(i4), public mo_ncwrite::nvars
```

Referenced by `create_netcdf()`, `mo_set_netcdf_outputs::set_netcdf()`, `write_dynamic_netcdf()`, and `write_static_netcdf()`.

15.64.2.10 v

```
type(variable), dimension(:), allocatable, public mo_ncwrite::v
```

Referenced by `create_ncdf()`, `mo_set_ncdf_outputs::set_ncdf()`, `write_dynamic_ncdf()`, and `write_static_ncdf()`.

15.65 mo_ncdf Module Reference

NetCDF Fortran 90 interface wrapper.

Data Types

- interface `ncdataset`
Provides basic file modification functionality.
- type `ncdimension`
Provides the dimension access functionality.
- interface `ncvariable`

Functions/Subroutines

- subroutine `initncvariable` (self, id, parent)
- subroutine `initncdimension` (self, id, parent)
- subroutine `initncdataset` (self, fname, mode)
- type(`ncvariable`) function `newncvariable` (id, parent)
- type(`ncdimension`) function `newncdimension` (id, parent)
- type(`ncdataset`) function `newncdataset` (fname, mode)
- subroutine `close` (self)
- integer(i4) function `getnvariables` (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 `getvariablelename` (self)
- integer(i4) function `getnodimensions` (self)
- type(`ncdimension`) function, dimension(:), allocatable `getvariableldimensions` (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 `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)

15.65.1 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

15.65.2 Function/Subroutine Documentation

15.65.2.1 `check()`

```
subroutine mo_netcdf::check (
    integer(i4), intent(in) status,
    character(*), intent(in) msg )
```

Referenced by `close()`, `getdata1df32()`, `getdata1df64()`, `getdata1di32()`, `getdata1di64()`, `getdata1di8()`, `getdata2df32()`, `getdata2df64()`, `getdata2di16()`, `getdata2di32()`, `getdata2di64()`, `getdata2di8()`, `getdata3df32()`, `getdata3df64()`, `getdata3di16()`, `getdata3di32()`, `getdata3di64()`, `getdata3di8()`, `getdata4df32()`, `getdata4df64()`, `getdata4di16()`, `getdata4di32()`, `getdata4di64()`, `getdata4di8()`, `getdata5df32()`, `getdata5df64()`, `getdata5di16()`, `getdata5di32()`, `getdata5di64()`, `getdata5di8()`, `getdatascalarf32()`, `getdatascalarf64()`, `getdatascalari16()`, `getdatascalari32()`, `getdatascalari64()`, `getdatascalari8()`, `getdimensionbyid()`, `getdimensionbyname()`, `getdimensionlength()`, `getdimensionname()`, `getglobalattributechar()`, `getglobalattributef32()`, `getglobalattributef64()`, `getglobalattributei16()`, `getglobalattributei32()`, `getglobalattributei64()`, `getglobalattributei8()`, `getnondimensions()`, `getnovariables()`, `getunlimiteddimension()`, `getvariableattributechar()`, `getvariableattributef32()`, `getvariableattributef64()`, `getvariableattributei16()`, `getvariableattributei32()`, `getvariableattributei64()`, `getvariableattributei8()`, `getvariablebyname()`, `getvariabledimensions()`, `getvariabledtype()`, `getvariableids()`, `getvariablename()`, `initncdataset()`, `isdatasetunlimited()`, `setdata1df32()`, `setdata1df64()`, `setdata1di16()`, `setdata1di32()`, `setdata1di64()`, `setdata1di8()`, `setdata2df32()`, `setdata2df64()`, `setdata2di16()`, `setdata2di32()`, `setdata2di64()`, `setdata2di8()`, `setdata3df32()`, `setdata3df64()`, `setdata3di16()`, `setdata3di32()`, `setdata3di64()`, `setdata3di8()`, `setdata4df32()`, `setdata4df64()`, `setdata4di16()`, `setdata4di32()`, `setdata4di64()`, `setdata4di8()`, `setdata5df32()`, `setdata5df64()`, `setdata5di16()`, `setdata5di32()`, `setdata5di64()`, `setdata5di8()`, `setdatascalarf32()`, `setdatascalarf64()`, `setdatascalari16()`, `setdatascalari32()`, `setdatascalari64()`, `setdatascalari8()`, `setdimension()`, `setglobalattributechar()`, `setglobalattributef32()`, `setglobalattributef64()`, `setglobalattributei16()`, `setglobalattributei32()`, `setglobalattributei64()`, `setglobalattributei8()`, `setvariableattributechar()`, `setvariableattributef32()`, `setvariableattributef64()`, `setvariableattributei16()`, `setvariableattributei32()`, `setvariableattributei64()`, `setvariableattributei8()`, and `setvariablewithids()`.

15.65.2.2 close()

```
subroutine mo_ncdf::close (
    class(ncdataset) self )
```

References check().

Here is the call graph for this function:



15.65.2.3 equalncdimensions()

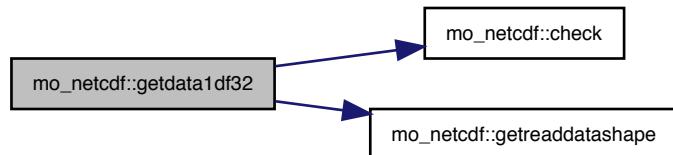
```
logical function mo_ncdf::equalncdimensions (
    type(ncdimension), intent(in) dim1,
    type(ncdimension), intent(in) dim2 )
```

15.65.2.4 getdata1df32()

```
subroutine mo_ncdf::getdata1df32 (
    class(ncvariable), intent(in) self,
    real(sp), dimension(:), intent(out), allocatable data,
    integer(i4), dimension(:), intent(in), optional start,
    integer(i4), dimension(:), intent(in), optional cnt,
    integer(i4), dimension(:), intent(in), optional stride,
    integer(i4), dimension(:), intent(in), optional map )
```

References check(), and getreaddatashape().

Here is the call graph for this function:

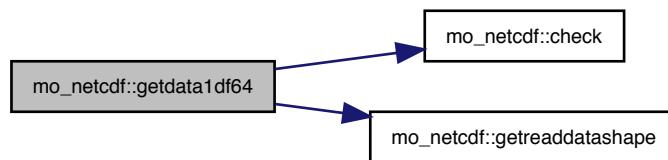


15.65.2.5 getdata1df64()

```
subroutine mo_netcdf::getdata1df64 (
    class(ncvariable), intent(in) self,
    real(dp), dimension(:), intent(out), allocatable data,
    integer(i4), dimension(:), intent(in), optional start,
    integer(i4), dimension(:), intent(in), optional cnt,
    integer(i4), dimension(:), intent(in), optional stride,
    integer(i4), dimension(:), intent(in), optional map )
```

References check(), and getreaddatashape().

Here is the call graph for this function:

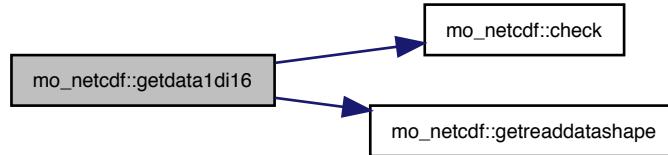


15.65.2.6 getdata1di16()

```
subroutine mo_netcdf::getdata1di16 (
    class(ncvariable), intent(in) self,
    integer(i2), dimension(:), intent(out), allocatable data,
    integer(i4), dimension(:), intent(in), optional start,
    integer(i4), dimension(:), intent(in), optional cnt,
    integer(i4), dimension(:), intent(in), optional stride,
    integer(i4), dimension(:), intent(in), optional map )
```

References check(), and getreaddatashape().

Here is the call graph for this function:

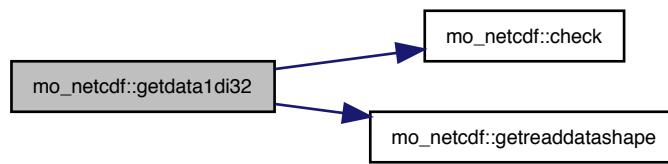


15.65.2.7 getdata1di32()

```
subroutine mo_netcdf::getdata1di32 (
    class(ncvariable), intent(in) self,
    integer(i4), dimension(:), intent(out), allocatable data,
    integer(i4), dimension(:), intent(in), optional start,
    integer(i4), dimension(:), intent(in), optional cnt,
    integer(i4), dimension(:), intent(in), optional stride,
    integer(i4), dimension(:), intent(in), optional map )
```

References check(), and getreaddatashape().

Here is the call graph for this function:

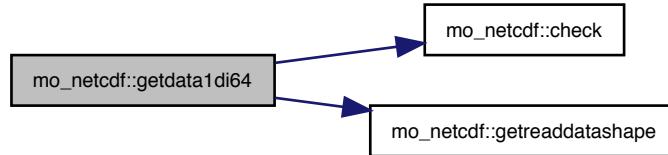


15.65.2.8 getdata1di64()

```
subroutine mo_netcdf::getdata1di64 (
    class(ncvariable), intent(in) self,
    integer(i8), dimension(:), intent(out), allocatable data,
    integer(i4), dimension(:), intent(in), optional start,
    integer(i4), dimension(:), intent(in), optional cnt,
    integer(i4), dimension(:), intent(in), optional stride,
    integer(i4), dimension(:), intent(in), optional map )
```

References check(), and getreaddatashape().

Here is the call graph for this function:

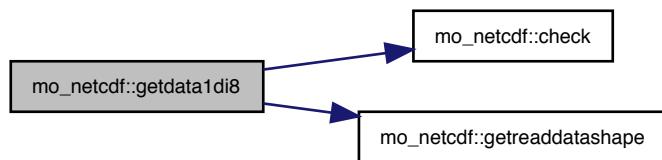


15.65.2.9 getdata1di8()

```
subroutine mo_netcdf::getdata1di8 (
    class(ncvariable), intent(in) self,
    integer(i1), dimension(:), intent(out), allocatable data,
    integer(i4), dimension(:), intent(in), optional start,
    integer(i4), dimension(:), intent(in), optional cnt,
    integer(i4), dimension(:), intent(in), optional stride,
    integer(i4), dimension(:), intent(in), optional map )
```

References check(), and getreaddatashape().

Here is the call graph for this function:

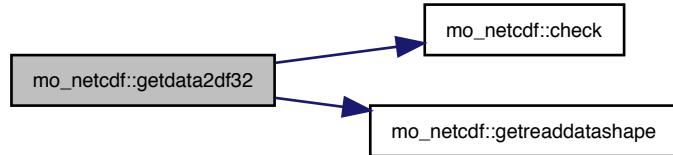


15.65.2.10 getdata2df32()

```
subroutine mo_netcdf::getdata2df32 (
    class(ncvariable), intent(in) self,
    real(sp), dimension(:, :), intent(out), allocatable data,
    integer(i4), dimension(:), intent(in), optional start,
    integer(i4), dimension(:), intent(in), optional cnt,
    integer(i4), dimension(:), intent(in), optional stride,
    integer(i4), dimension(:), intent(in), optional map )
```

References check(), and getreaddatashape().

Here is the call graph for this function:

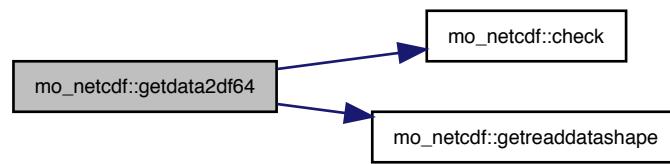


15.65.2.11 getdata2df64()

```
subroutine mo_netcdf::getdata2df64 (
    class(ncvariable), intent(in) self,
    real(dp), dimension(:, :), intent(out), allocatable data,
    integer(i4), dimension(:), intent(in), optional start,
    integer(i4), dimension(:), intent(in), optional cnt,
    integer(i4), dimension(:), intent(in), optional stride,
    integer(i4), dimension(:), intent(in), optional map )
```

References check(), and getreaddatashape().

Here is the call graph for this function:

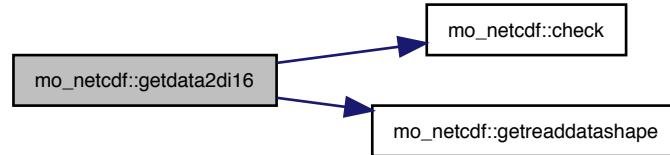


15.65.2.12 getdata2di16()

```
subroutine mo_netcdf::getdata2di16 (
    class(ncvariable), intent(in) self,
    integer(i2), dimension(:, :), intent(out), allocatable data,
    integer(i4), dimension(:), intent(in), optional start,
    integer(i4), dimension(:), intent(in), optional cnt,
    integer(i4), dimension(:), intent(in), optional stride,
    integer(i4), dimension(:), intent(in), optional map )
```

References check(), and getreaddatashape().

Here is the call graph for this function:

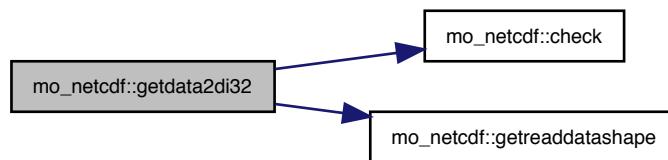


15.65.2.13 getdata2di32()

```
subroutine mo_netcdf::getdata2di32 (
    class(ncvariable), intent(in) self,
    integer(i4), dimension(:, :), intent(out), allocatable data,
    integer(i4), dimension(:), intent(in), optional start,
    integer(i4), dimension(:), intent(in), optional cnt,
    integer(i4), dimension(:), intent(in), optional stride,
    integer(i4), dimension(:), intent(in), optional map )
```

References check(), and getreaddatashape().

Here is the call graph for this function:

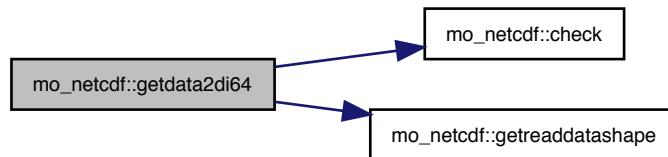


15.65.2.14 getdata2di64()

```
subroutine mo_netcdf::getdata2di64 (
    class(ncvariable), intent(in) self,
    integer(i8), dimension(:, :), intent(out), allocatable data,
    integer(i4), dimension(:), intent(in), optional start,
    integer(i4), dimension(:), intent(in), optional cnt,
    integer(i4), dimension(:), intent(in), optional stride,
    integer(i4), dimension(:), intent(in), optional map )
```

References check(), and getreaddatashape().

Here is the call graph for this function:

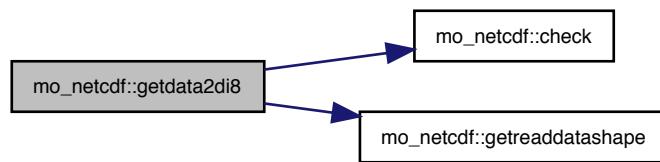


15.65.2.15 getdata2di8()

```
subroutine mo_ncdf::getdata2di8 (
    class(ncvariable), intent(in) self,
    integer(i1), dimension(:, :, ), intent(out), allocatable data,
    integer(i4), dimension(:, ), intent(in), optional start,
    integer(i4), dimension(:, ), intent(in), optional cnt,
    integer(i4), dimension(:, ), intent(in), optional stride,
    integer(i4), dimension(:, ), intent(in), optional map )
```

References check(), and getreaddatashape().

Here is the call graph for this function:

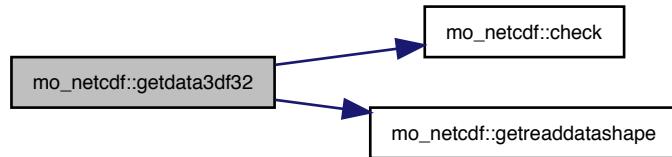


15.65.2.16 getdata3df32()

```
subroutine mo_ncdf::getdata3df32 (
    class(ncvariable), intent(in) self,
    real(sp), dimension(:, :, :, ), intent(out), allocatable data,
    integer(i4), dimension(:, ), intent(in), optional start,
    integer(i4), dimension(:, ), intent(in), optional cnt,
    integer(i4), dimension(:, ), intent(in), optional stride,
    integer(i4), dimension(:, ), intent(in), optional map )
```

References check(), and getreaddatashape().

Here is the call graph for this function:

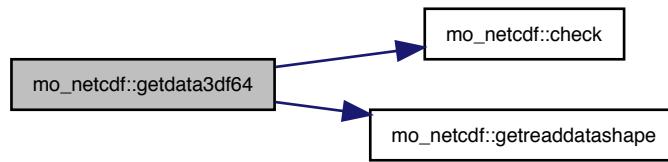


15.65.2.17 getdata3df64()

```
subroutine mo_netcdf::getdata3df64 (
    class(ncvariable), intent(in) self,
    real(dp), dimension(:, :, :), intent(out), allocatable data,
    integer(i4), dimension(:), intent(in), optional start,
    integer(i4), dimension(:), intent(in), optional cnt,
    integer(i4), dimension(:), intent(in), optional stride,
    integer(i4), dimension(:), intent(in), optional map )
```

References check(), and getreaddatashape().

Here is the call graph for this function:

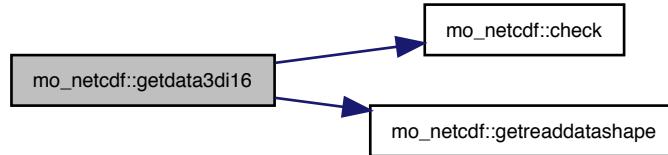


15.65.2.18 getdata3di16()

```
subroutine mo_netcdf::getdata3di16 (
    class(ncvariable), intent(in) self,
    integer(i2), dimension(:, :, :), intent(out), allocatable data,
    integer(i4), dimension(:), intent(in), optional start,
    integer(i4), dimension(:), intent(in), optional cnt,
    integer(i4), dimension(:), intent(in), optional stride,
    integer(i4), dimension(:), intent(in), optional map )
```

References check(), and getreaddatashape().

Here is the call graph for this function:

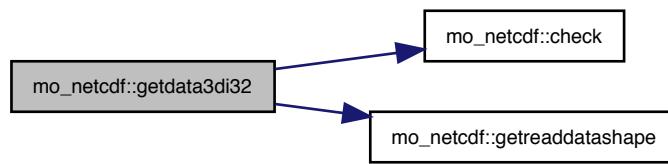


15.65.2.19 getdata3di32()

```
subroutine mo_netcdf::getdata3di32 (
    class(ncvariable), intent(in) self,
    integer(i4), dimension(:, :, :), intent(out), allocatable data,
    integer(i4), dimension(:), intent(in), optional start,
    integer(i4), dimension(:), intent(in), optional cnt,
    integer(i4), dimension(:), intent(in), optional stride,
    integer(i4), dimension(:), intent(in), optional map )
```

References check(), and getreaddatashape().

Here is the call graph for this function:

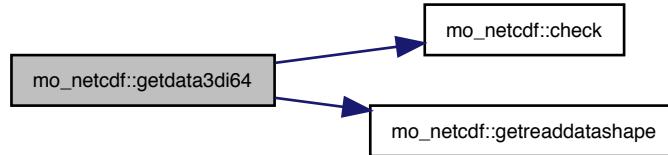


15.65.2.20 getdata3di64()

```
subroutine mo_netcdf::getdata3di64 (
    class(ncvariable), intent(in) self,
    integer(i8), dimension(:, :, :), intent(out), allocatable data,
    integer(i4), dimension(:), intent(in), optional start,
    integer(i4), dimension(:), intent(in), optional cnt,
    integer(i4), dimension(:), intent(in), optional stride,
    integer(i4), dimension(:), intent(in), optional map )
```

References check(), and getreaddatashape().

Here is the call graph for this function:

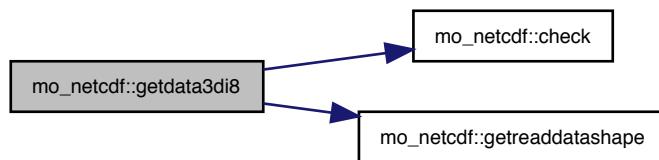


15.65.2.21 getdata3di8()

```
subroutine mo_netcdf::getdata3di8 (
    class(ncvariable), intent(in) self,
    integer(i1), dimension(:, :, :), intent(out), allocatable data,
    integer(i4), dimension(:), intent(in), optional start,
    integer(i4), dimension(:), intent(in), optional cnt,
    integer(i4), dimension(:), intent(in), optional stride,
    integer(i4), dimension(:), intent(in), optional map )
```

References check(), and getreaddatashape().

Here is the call graph for this function:

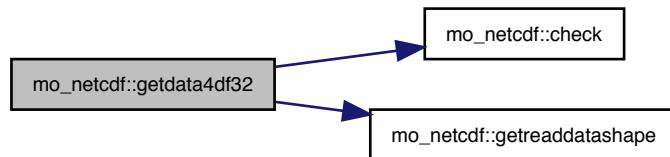


15.65.2.22 getdata4df32()

```
subroutine mo_netcdf::getdata4df32 (
    class(ncvariable), intent(in) self,
    real(sp), dimension(:, :, :, :), intent(out), allocatable data,
    integer(i4), dimension(:), intent(in), optional start,
    integer(i4), dimension(:), intent(in), optional cnt,
    integer(i4), dimension(:), intent(in), optional stride,
    integer(i4), dimension(:), intent(in), optional map )
```

References check(), and getreaddatashape().

Here is the call graph for this function:

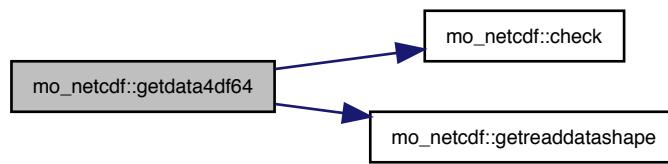


15.65.2.23 getdata4df64()

```
subroutine mo_netcdf::getdata4df64 (
    class(ncvariable), intent(in) self,
    real(dp), dimension(:, :, :, :), intent(out), allocatable data,
    integer(i4), dimension(:), intent(in), optional start,
    integer(i4), dimension(:), intent(in), optional cnt,
    integer(i4), dimension(:), intent(in), optional stride,
    integer(i4), dimension(:), intent(in), optional map )
```

References check(), and getreaddatashape().

Here is the call graph for this function:

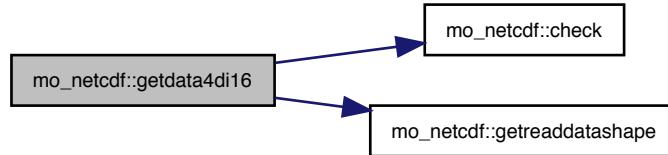


15.65.2.24 getdata4di16()

```
subroutine mo_netcdf::getdata4di16 (
    class(ncvariable), intent(in) self,
    integer(i2), dimension(:, :, :, :), intent(out), allocatable data,
    integer(i4), dimension(:), intent(in), optional start,
    integer(i4), dimension(:), intent(in), optional cnt,
    integer(i4), dimension(:), intent(in), optional stride,
    integer(i4), dimension(:), intent(in), optional map )
```

References check(), and getreaddatashape().

Here is the call graph for this function:

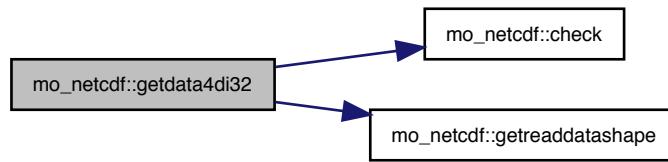


15.65.2.25 getdata4di32()

```
subroutine mo_netcdf::getdata4di32 (
    class(ncvariable), intent(in) self,
    integer(i4), dimension(:, :, :, :), intent(out), allocatable data,
    integer(i4), dimension(:), intent(in), optional start,
    integer(i4), dimension(:), intent(in), optional cnt,
    integer(i4), dimension(:), intent(in), optional stride,
    integer(i4), dimension(:), intent(in), optional map )
```

References check(), and getreaddatashape().

Here is the call graph for this function:

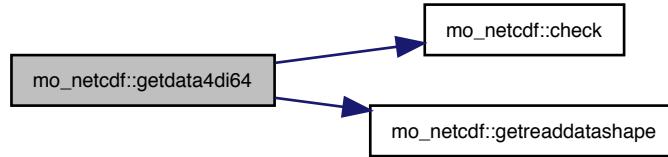


15.65.2.26 getdata4di64()

```
subroutine mo_netcdf::getdata4di64 (
    class(ncvariable), intent(in) self,
    integer(i8), dimension(:, :, :, :), intent(out), allocatable data,
    integer(i4), dimension(:), intent(in), optional start,
    integer(i4), dimension(:), intent(in), optional cnt,
    integer(i4), dimension(:), intent(in), optional stride,
    integer(i4), dimension(:), intent(in), optional map )
```

References check(), and getreaddatashape().

Here is the call graph for this function:

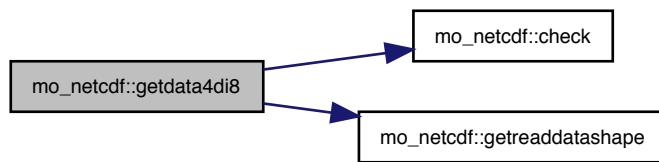


15.65.2.27 getdata4di8()

```
subroutine mo_netcdf::getdata4di8 (
    class(ncvariable), intent(in) self,
    integer(i1), dimension(:, :, :, :), intent(out), allocatable data,
    integer(i4), dimension(:), intent(in), optional start,
    integer(i4), dimension(:), intent(in), optional cnt,
    integer(i4), dimension(:), intent(in), optional stride,
    integer(i4), dimension(:), intent(in), optional map )
```

References check(), and getreaddatashape().

Here is the call graph for this function:

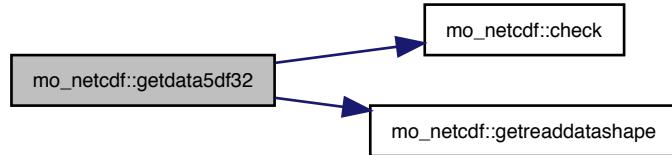


15.65.2.28 getdata5df32()

```
subroutine mo_netcdf::getdata5df32 (
    class(ncvariable), intent(in) self,
    real(sp), dimension(:, :, :, :, :), intent(out), allocatable data,
    integer(i4), dimension(:), intent(in), optional start,
    integer(i4), dimension(:), intent(in), optional cnt,
    integer(i4), dimension(:), intent(in), optional stride,
    integer(i4), dimension(:), intent(in), optional map )
```

References check(), and getreaddatashape().

Here is the call graph for this function:

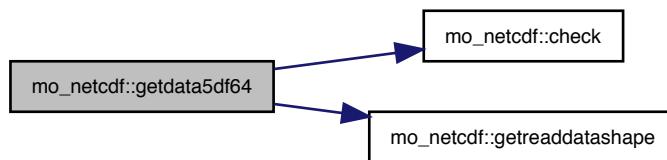


15.65.2.29 getdata5df64()

```
subroutine mo_netcdf::getdata5df64 (
    class(ncvariable), intent(in) self,
    real(dp), dimension(:, :, :, :, :), intent(out), allocatable data,
    integer(i4), dimension(:), intent(in), optional start,
    integer(i4), dimension(:), intent(in), optional cnt,
    integer(i4), dimension(:), intent(in), optional stride,
    integer(i4), dimension(:), intent(in), optional map )
```

References check(), and getreaddatashape().

Here is the call graph for this function:

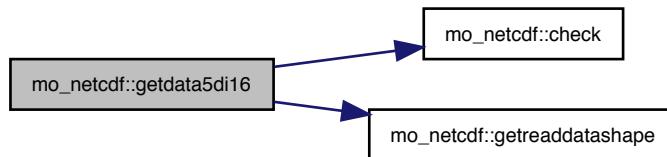


15.65.2.30 getdata5di16()

```
subroutine mo_netcdf::getdata5di16 (
    class(ncvariable), intent(in) self,
    integer(i2), dimension(:, :, :, :, :), intent(out), allocatable data,
    integer(i4), dimension(:), intent(in), optional start,
    integer(i4), dimension(:), intent(in), optional cnt,
    integer(i4), dimension(:), intent(in), optional stride,
    integer(i4), dimension(:), intent(in), optional map )
```

References check(), and getreaddatashape().

Here is the call graph for this function:

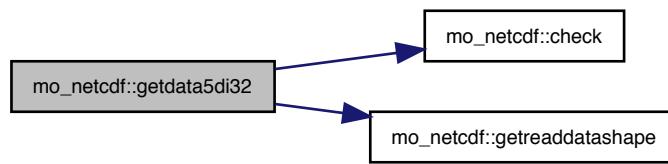


15.65.2.31 getdata5di32()

```
subroutine mo_netcdf::getdata5di32 (
    class(ncvariable), intent(in) self,
    integer(i4), dimension(:, :, :, :, :), intent(out), allocatable data,
    integer(i4), dimension(:), intent(in), optional start,
    integer(i4), dimension(:), intent(in), optional cnt,
    integer(i4), dimension(:), intent(in), optional stride,
    integer(i4), dimension(:), intent(in), optional map )
```

References check(), and getreaddatashape().

Here is the call graph for this function:

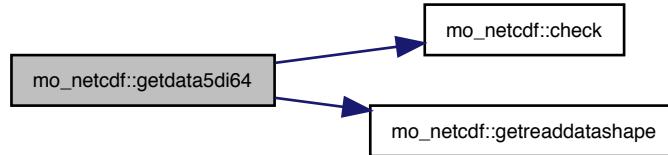


15.65.2.32 getdata5di64()

```
subroutine mo_netcdf::getdata5di64 (
    class(ncvariable), intent(in) self,
    integer(i8), dimension(:, :, :, :, :), intent(out), allocatable data,
    integer(i4), dimension(:), intent(in), optional start,
    integer(i4), dimension(:), intent(in), optional cnt,
    integer(i4), dimension(:), intent(in), optional stride,
    integer(i4), dimension(:), intent(in), optional map )
```

References check(), and getreaddatashape().

Here is the call graph for this function:

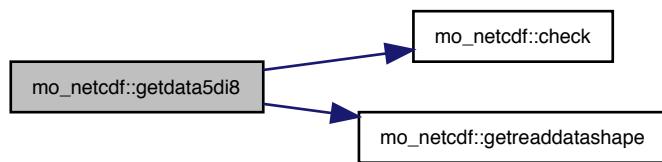


15.65.2.33 getdata5di8()

```
subroutine mo_netcdf::getdata5di8 (
    class(ncvariable), intent(in) self,
    integer(i1), dimension(:, :, :, :, :), intent(out), allocatable data,
    integer(i4), dimension(:), intent(in), optional start,
    integer(i4), dimension(:), intent(in), optional cnt,
    integer(i4), dimension(:), intent(in), optional stride,
    integer(i4), dimension(:), intent(in), optional map )
```

References `check()`, and `getreaddatashape()`.

Here is the call graph for this function:

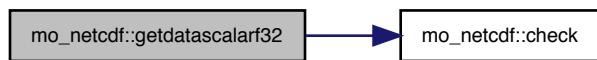


15.65.2.34 getdatascalarf32()

```
subroutine mo_netcdf::getdatascalarf32 (
    class(ncvariable), intent(in) self,
    real(sp), intent(out) data,
    integer(i4), dimension(:), intent(in), optional start,
    integer(i4), dimension(:), intent(in), optional cnt,
    integer(i4), dimension(:), intent(in), optional stride,
    integer(i4), dimension(:), intent(in), optional map )
```

References `check()`.

Here is the call graph for this function:



15.65.2.35 getdatascalarf64()

```
subroutine mo_netcdf::getdatascalarf64 (
```

```
class(ncvariable), intent(in) self,
real(dp), intent(out) data,
integer(i4), dimension(:), intent(in), optional start,
integer(i4), dimension(:), intent(in), optional cnt,
integer(i4), dimension(:), intent(in), optional stride,
integer(i4), dimension(:), intent(in), optional map )
```

References check().

Here is the call graph for this function:



15.65.2.36 getdatascalari16()

```
subroutine mo_ncdf::getdatascalari16 (
    class(ncvariable), intent(in) self,
    integer(i2), intent(out) data,
    integer(i4), dimension(:), intent(in), optional start,
    integer(i4), dimension(:), intent(in), optional cnt,
    integer(i4), dimension(:), intent(in), optional stride,
    integer(i4), dimension(:), intent(in), optional map )
```

References check().

Here is the call graph for this function:

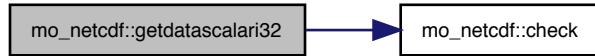


15.65.2.37 getdatascalari32()

```
subroutine mo_ncdf::getdatascalari32 (
    class(ncvariable), intent(in) self,
    integer(i4), intent(out) data,
    integer(i4), dimension(:), intent(in), optional start,
    integer(i4), dimension(:), intent(in), optional cnt,
    integer(i4), dimension(:), intent(in), optional stride,
    integer(i4), dimension(:), intent(in), optional map )
```

References check().

Here is the call graph for this function:



15.65.2.38 getdatascalari64()

```
subroutine mo_ncdf::getdatascalari64 (
    class(ncvariable), intent(in) self,
    integer(i8), intent(out) data,
    integer(i4), dimension(:), intent(in), optional start,
    integer(i4), dimension(:), intent(in), optional cnt,
    integer(i4), dimension(:), intent(in), optional stride,
    integer(i4), dimension(:), intent(in), optional map )
```

References check().

Here is the call graph for this function:



15.65.2.39 getdatascalari8()

```
subroutine mo_ncdf::getdatascalari8 (
    class(ncvariable), intent(in) self,
    integer(i1), intent(out) data,
    integer(i4), dimension(:), intent(in), optional start,
    integer(i4), dimension(:), intent(in), optional cnt,
    integer(i4), dimension(:), intent(in), optional stride,
    integer(i4), dimension(:), intent(in), optional map )
```

References check().

Here is the call graph for this function:



15.65.2.40 getdimensionbyid()

```
type(ncdimension) function mo_ncdf::getdimensionbyid (
    class(ncdataset), intent(in) self,
    integer(i4) id )
```

References check().

Here is the call graph for this function:



15.65.2.41 getdimensionbyname()

```
type(ncdimension) function mo_ncdf::getdimensionbyname (
    class(ncdataset), intent(in) self,
    character(*) name )
```

References check().

Here is the call graph for this function:



15.65.2.42 getdimensionlength()

```
integer(i4) function mo_netcdf::getdimensionlength (
    class(ncdimension), intent(in) self )
```

References check().

Here is the call graph for this function:

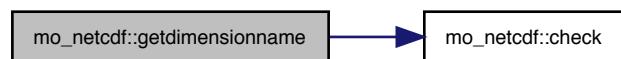


15.65.2.43 getdimensionname()

```
character(len = 256) function mo_netcdf::getdimensionname (
    class(ncdimension), intent(in) self )
```

References check().

Here is the call graph for this function:

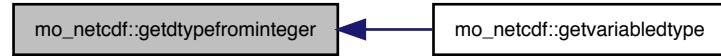


15.65.2.44 getdtypefrominteger()

```
character(3) function mo_netcdf::getdtypefrominteger (
    integer(i4) dtype )
```

Referenced by getvariabledtype().

Here is the caller graph for this function:



15.65.2.45 getdatatypefromstring()

```
integer(i4) function mo_ncdf::getdatatypefromstring (
    character(*) dtype )
```

Referenced by `setvariablewithids()`.

Here is the caller graph for this function:



15.65.2.46 getglobalattributechar()

```
subroutine mo_ncdf::getglobalattributechar (
    class(ncdataset), intent(in) self,
    character(*), intent(in) name,
    character(*), intent(out) avalue )
```

References `check()`.

Here is the call graph for this function:



15.65.2.47 getglobalattributef32()

```
subroutine mo_netcdf::getglobalattributef32 (
    class(ncdataset), intent(in) self,
    character(*), intent(in) name,
    real(sp), intent(out) avalue )
```

References check().

Here is the call graph for this function:

**15.65.2.48 getglobalattributef64()**

```
subroutine mo_netcdf::getglobalattributef64 (
    class(ncdataset), intent(in) self,
    character(*), intent(in) name,
    real(dp), intent(out) avalue )
```

References check().

Here is the call graph for this function:

**15.65.2.49 getglobalattributei16()**

```
subroutine mo_netcdf::getglobalattributei16 (
    class(ncdataset), intent(in) self,
    character(*), intent(in) name,
    integer(i2), intent(out) avalue )
```

References check().

Here is the call graph for this function:



15.65.2.50 getglobalattributei32()

```
subroutine mo_ncdf::getglobalattributei32 (
    class(ncdataset), intent(in) self,
    character(*), intent(in) name,
    integer(i4), intent(out) avalue )
```

References check().

Here is the call graph for this function:



15.65.2.51 getglobalattributei64()

```
subroutine mo_ncdf::getglobalattributei64 (
    class(ncdataset), intent(in) self,
    character(*), intent(in) name,
    integer(i8), intent(out) avalue )
```

References check().

Here is the call graph for this function:



15.65.2.52 getglobalattributei8()

```
subroutine mo_netcdf::getglobalattributei8 (
    class(ncdataset), intent(in) self,
    character(*), intent(in) name,
    integer(i1), intent(out) avalue )
```

References check().

Here is the call graph for this function:



15.65.2.53 getnordimensions()

```
integer(i4) function mo_netcdf::getnordimensions (
    class(ncvariable), intent(in) self )
```

References check().

Here is the call graph for this function:



15.65.2.54 getnvariables()

```
integer(i4) function mo_netcdf::getnvariables (
    class(ncdataset), intent(in) self )
```

References check().

Here is the call graph for this function:

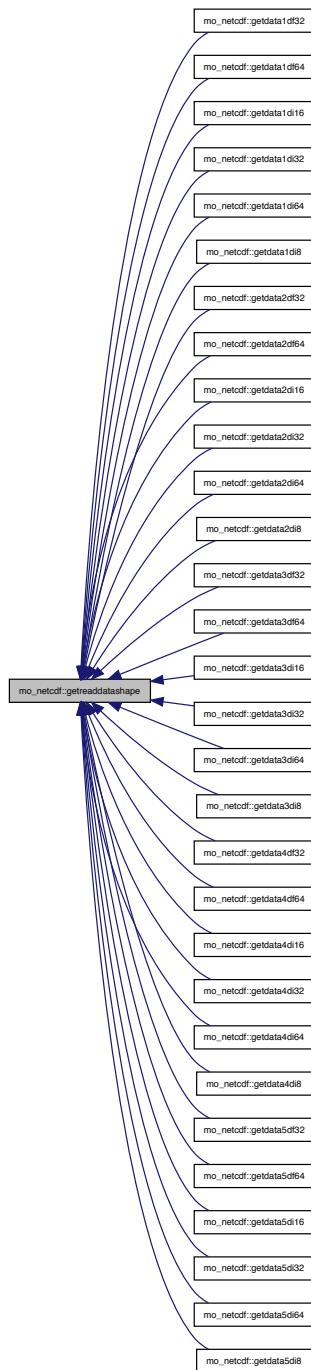


15.65.2.55 getreaddatashape()

```
integer(i4) function, dimension(datarank) mo_ncdf::getreaddatashape (
    type(ncvariable), intent(in) var,
    integer(i4), intent(in) datarank,
    integer(i4), dimension(:), intent(in), optional instart,
    integer(i4), dimension(:), intent(in), optional incnt,
    integer(i4), dimension(:), intent(in), optional instride )
```

Referenced by `getdata1df32()`, `getdata1df64()`, `getdata1di16()`, `getdata1di32()`, `getdata1di64()`, `getdata1di8()`, `getdata2df32()`, `getdata2df64()`, `getdata2di16()`, `getdata2di32()`, `getdata2di64()`, `getdata2di8()`, `getdata3df32()`, `getdata3df64()`, `getdata3di16()`, `getdata3di32()`, `getdata3di64()`, `getdata3di8()`, `getdata4df32()`, `getdata4df64()`, `getdata4di16()`, `getdata4di32()`, `getdata4di64()`, `getdata4di8()`, `getdata5df32()`, `getdata5df64()`, `getdata5di16()`, `getdata5di32()`, `getdata5di64()`, and `getdata5di8()`.

Here is the caller graph for this function:



15.65.2.56 getunlimiteddimension()

```
type(ncdimension) function mo_netcdf::getunlimiteddimension (
```

```
    class(ncdataset), intent(in) self )
```

References check().

Here is the call graph for this function:



15.65.2.57 getvariableattributechar()

```
subroutine mo_ncdf::getvariableattributechar (
    class(ncvariable), intent(in) self,
    character(*), intent(in) name,
    character(*), intent(out) avalue )
```

References check().

Here is the call graph for this function:



15.65.2.58 getvariableattributef32()

```
subroutine mo_ncdf::getvariableattributef32 (
    class(ncvariable), intent(in) self,
    character(*), intent(in) name,
    real(sp), intent(out) avalue )
```

References check().

Here is the call graph for this function:



15.65.2.59 getvariableattributef64()

```
subroutine mo_netcdf::getvariableattributef64 (
    class(ncvariable), intent(in) self,
    character(*), intent(in) name,
    real(dp), intent(out) avalue )
```

References check().

Here is the call graph for this function:



15.65.2.60 getvariableattributei16()

```
subroutine mo_netcdf::getvariableattributei16 (
    class(ncvariable), intent(in) self,
    character(*), intent(in) name,
    integer(i2), intent(out) avalue )
```

References check().

Here is the call graph for this function:



15.65.2.61 getvariableattributei32()

```
subroutine mo_netcdf::getvariableattributei32 (
    class(ncvariable), intent(in) self,
    character(*), intent(in) name,
    integer(i4), intent(out) avalue )
```

References check().

Here is the call graph for this function:



15.65.2.62 getvariableattributei64()

```
subroutine mo_ncdf::getvariableattributei64 (
    class(ncvariable), intent(in) self,
    character(*), intent(in) name,
    integer(i8), intent(out) avalue )
```

References check().

Here is the call graph for this function:



15.65.2.63 getvariableattributei8()

```
subroutine mo_ncdf::getvariableattributei8 (
    class(ncvariable), intent(in) self,
    character(*), intent(in) name,
    integer(i1), intent(out) avalue )
```

References check().

Here is the call graph for this function:



15.65.2.64 getvariablebyname()

```
type(ncvariable) function mo_netcdf::getvariablebyname (
    class(ncdataset), intent(in) self,
    character(*), intent(in) name )
```

References check().

Here is the call graph for this function:



15.65.2.65 getvariabledimensions()

```
type(ncdimension) function, dimension(:), allocatable mo_netcdf::getvariabledimensions (
    class(ncvariable), intent(in) self )
```

References check().

Here is the call graph for this function:

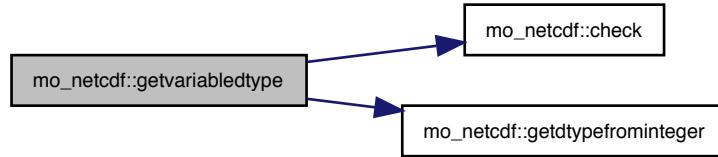


15.65.2.66 getvariabledtype()

```
character(3) function mo_netcdf::getvariabledtype (
    class(ncvariable), intent(in) self )
```

References check(), and getdtypefrominteger().

Here is the call graph for this function:



15.65.2.67 getvariablefillvaluef32()

```
subroutine mo_ncdf::getvariablefillvaluef32 (
    class(ncvariable), intent(in) self,
    real(sp), intent(out) fvalue )
```

15.65.2.68 getvariablefillvaluef64()

```
subroutine mo_ncdf::getvariablefillvaluef64 (
    class(ncvariable), intent(in) self,
    real(dp), intent(out) fvalue )
```

15.65.2.69 getvariablefillvaluei16()

```
subroutine mo_ncdf::getvariablefillvaluei16 (
    class(ncvariable), intent(in) self,
    integer(i2), intent(out) fvalue )
```

15.65.2.70 getvariablefillvaluei32()

```
subroutine mo_ncdf::getvariablefillvaluei32 (
    class(ncvariable), intent(in) self,
    integer(i4), intent(out) fvalue )
```

15.65.2.71 getvariablefillvaluei64()

```
subroutine mo_ncdf::getvariablefillvaluei64 (
    class(ncvariable), intent(in) self,
    integer(i8), intent(out) fvalue )
```

15.65.2.72 getvariablefillvaluei8()

```
subroutine mo_netcdf::getvariablefillvaluei8 (
    class(ncvariable), intent(in) self,
    integer(i1), intent(out) fvalue )
```

15.65.2.73 getvariableids()

```
integer(i4) function, dimension(:), allocatable mo_netcdf::getvariableids (
    class(ncdataset), intent(in) self )
```

References check().

Here is the call graph for this function:

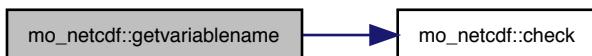


15.65.2.74 getvariablename()

```
character(len = 256) function mo_netcdf::getvariablename (
    class(ncvariable), intent(in) self )
```

References check().

Here is the call graph for this function:



15.65.2.75 getvariables()

```
type(ncvariable) function, dimension(:), allocatable mo_netcdf::getvariables (
    class(ncdataset), intent(in) self )
```

15.65.2.76 getvariablesshape()

```
integer(i4) function, dimension(:), allocatable mo_ncdf::getvariablesshape (  
    class(ncvariable), intent(in) self )
```

15.65.2.77 hasattribute()

```
logical function mo_ncdf::hasattribute (  
    class(ncvariable), intent(in) self,  
    character(*), intent(in) name )
```

15.65.2.78 hasdimension()

```
logical function mo_ncdf::hasdimension (  
    class(ncdataset), intent(in) self,  
    character(*), intent(in) name )
```

15.65.2.79 hasvariable()

```
logical function mo_ncdf::hasvariable (  
    class(ncdataset), intent(in) self,  
    character(*), intent(in) name )
```

15.65.2.80 initncdataset()

```
subroutine mo_ncdf::initncdataset (  
    class(ncdataset), intent(inout) self,  
    character(*), intent(in) fname,  
    character(1), intent(in) mode )
```

References check().

Here is the call graph for this function:

**15.65.2.81 initncdimension()**

```
subroutine mo_ncdf::initncdimension (
```

```
class(ncdimension), intent(inout) self,
integer(i4), intent(in) id,
type(ncdataset), intent(in) parent )
```

15.65.2.82 initncvariable()

```
subroutine mo_netcdf::initncvariable (
    class(ncvariable), intent(inout) self,
    integer(i4), intent(in) id,
    type(ncdataset), intent(in) parent )
```

15.65.2.83 isdatasetunlimited()

```
logical function mo_netcdf::isdatasetunlimited (
    class(ncdataset), intent(in) self )
```

References check().

Here is the call graph for this function:



15.65.2.84 isunlimiteddimension()

```
logical function mo_netcdf::isunlimiteddimension (
    class(ncdimension), intent(in) self )
```

15.65.2.85 isunlimitedvariable()

```
logical function mo_netcdf::isunlimitedvariable (
    class(ncvariable), intent(in) self )
```

15.65.2.86 newncdataset()

```
type(ncdataset) function mo_netcdf::newncdataset (
    character(*), intent(in) fname,
    character(1), intent(in) mode )
```

15.65.2.87 newncdimension()

```
type(ncdimension) function mo_ncdf::newncdimension (
    integer(i4), intent(in) id,
    type(ncdataset), intent(in) parent )
```

15.65.2.88 newncvariable()

```
type(ncvariable) function mo_ncdf::newncvariable (
    integer(i4), intent(in) id,
    type(ncdataset), intent(in) parent )
```

15.65.2.89 setdata1df32()

```
subroutine mo_ncdf::setdata1df32 (
    class(ncvariable), intent(in) self,
    real(sp), dimension(:), intent(in) values,
    integer(i4), dimension(:), intent(in), optional start,
    integer(i4), dimension(:), intent(in), optional cnt,
    integer(i4), dimension(:), intent(in), optional stride,
    integer(i4), dimension(:), intent(in), optional map )
```

References check().

Here is the call graph for this function:

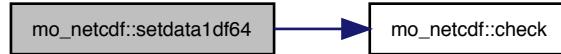


15.65.2.90 setdata1df64()

```
subroutine mo_ncdf::setdata1df64 (
    class(ncvariable), intent(in) self,
    real(dp), dimension(:), intent(in) values,
    integer(i4), dimension(:), intent(in), optional start,
    integer(i4), dimension(:), intent(in), optional cnt,
    integer(i4), dimension(:), intent(in), optional stride,
    integer(i4), dimension(:), intent(in), optional map )
```

References check().

Here is the call graph for this function:



15.65.2.91 setdata1di16()

```
subroutine mo_ncdf::setdata1di16 (
    class(ncvariable), intent(in) self,
    integer(i2), dimension(:), intent(in) values,
    integer(i4), dimension(:), intent(in), optional start,
    integer(i4), dimension(:), intent(in), optional cnt,
    integer(i4), dimension(:), intent(in), optional stride,
    integer(i4), dimension(:), intent(in), optional map )
```

References check().

Here is the call graph for this function:



15.65.2.92 setdata1di32()

```
subroutine mo_ncdf::setdata1di32 (
    class(ncvariable), intent(in) self,
    integer(i4), dimension(:), intent(in) values,
    integer(i4), dimension(:), intent(in), optional start,
    integer(i4), dimension(:), intent(in), optional cnt,
    integer(i4), dimension(:), intent(in), optional stride,
    integer(i4), dimension(:), intent(in), optional map )
```

References check().

Here is the call graph for this function:

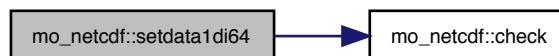


15.65.2.93 setdata1di64()

```
subroutine mo_ncdf::setdata1di64 (
    class(ncvariable), intent(in) self,
    integer(i8), dimension(:), intent(in) values,
    integer(i4), dimension(:), intent(in), optional start,
    integer(i4), dimension(:), intent(in), optional cnt,
    integer(i4), dimension(:), intent(in), optional stride,
    integer(i4), dimension(:), intent(in), optional map )
```

References check().

Here is the call graph for this function:

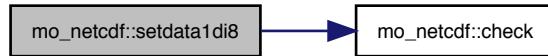


15.65.2.94 setdata1di8()

```
subroutine mo_ncdf::setdata1di8 (
    class(ncvariable), intent(in) self,
    integer(i1), dimension(:), intent(in) values,
    integer(i4), dimension(:), intent(in), optional start,
    integer(i4), dimension(:), intent(in), optional cnt,
    integer(i4), dimension(:), intent(in), optional stride,
    integer(i4), dimension(:), intent(in), optional map )
```

References check().

Here is the call graph for this function:



15.65.2.95 setdata2df32()

```
subroutine mo_ncdf::setdata2df32 (
    class(ncvariable), intent(in) self,
    real(sp), dimension(:, :, ), intent(in) values,
    integer(i4), dimension(:, ), intent(in), optional start,
    integer(i4), dimension(:, ), intent(in), optional cnt,
    integer(i4), dimension(:, ), intent(in), optional stride,
    integer(i4), dimension(:, ), intent(in), optional map )
```

References check().

Here is the call graph for this function:



15.65.2.96 setdata2df64()

```
subroutine mo_ncdf::setdata2df64 (
    class(ncvariable), intent(in) self,
    real(dp), dimension(:, :, ), intent(in) values,
    integer(i4), dimension(:, ), intent(in), optional start,
    integer(i4), dimension(:, ), intent(in), optional cnt,
    integer(i4), dimension(:, ), intent(in), optional stride,
    integer(i4), dimension(:, ), intent(in), optional map )
```

References check().

Here is the call graph for this function:

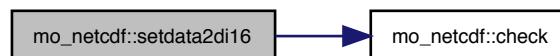


15.65.2.97 setdata2di16()

```
subroutine mo_ncdf::setdata2di16 (
    class(ncvariable), intent(in) self,
    integer(i2), dimension(:, :), intent(in) values,
    integer(i4), dimension(:, ), intent(in), optional start,
    integer(i4), dimension(:, ), intent(in), optional cnt,
    integer(i4), dimension(:, ), intent(in), optional stride,
    integer(i4), dimension(:, ), intent(in), optional map )
```

References check().

Here is the call graph for this function:

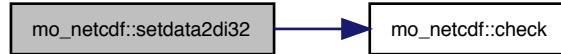


15.65.2.98 setdata2di32()

```
subroutine mo_ncdf::setdata2di32 (
    class(ncvariable), intent(in) self,
    integer(i4), dimension(:, :), intent(in) values,
    integer(i4), dimension(:, ), intent(in), optional start,
    integer(i4), dimension(:, ), intent(in), optional cnt,
    integer(i4), dimension(:, ), intent(in), optional stride,
    integer(i4), dimension(:, ), intent(in), optional map )
```

References check().

Here is the call graph for this function:



15.65.2.99 setdata2di64()

```
subroutine mo_ncdf::setdata2di64 (
    class(ncvariable), intent(in) self,
    integer(i8), dimension(:, :, ), intent(in) values,
    integer(i4), dimension(:, ), intent(in), optional start,
    integer(i4), dimension(:, ), intent(in), optional cnt,
    integer(i4), dimension(:, ), intent(in), optional stride,
    integer(i4), dimension(:, ), intent(in), optional map )
```

References check().

Here is the call graph for this function:

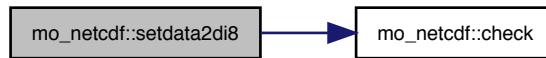


15.65.2.100 setdata2di8()

```
subroutine mo_ncdf::setdata2di8 (
    class(ncvariable), intent(in) self,
    integer(i1), dimension(:, :, ), intent(in) values,
    integer(i4), dimension(:, ), intent(in), optional start,
    integer(i4), dimension(:, ), intent(in), optional cnt,
    integer(i4), dimension(:, ), intent(in), optional stride,
    integer(i4), dimension(:, ), intent(in), optional map )
```

References check().

Here is the call graph for this function:

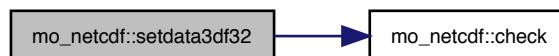


15.65.2.101 setdata3df32()

```
subroutine mo_ncdf::setdata3df32 (
    class(ncvariable), intent(in) self,
    real(sp), dimension(:, :, :), intent(in) values,
    integer(i4), dimension(:), intent(in), optional start,
    integer(i4), dimension(:), intent(in), optional cnt,
    integer(i4), dimension(:), intent(in), optional stride,
    integer(i4), dimension(:), intent(in), optional map )
```

References check().

Here is the call graph for this function:



15.65.2.102 setdata3df64()

```
subroutine mo_ncdf::setdata3df64 (
    class(ncvariable), intent(in) self,
    real(dp), dimension(:, :, :), intent(in) values,
    integer(i4), dimension(:), intent(in), optional start,
    integer(i4), dimension(:), intent(in), optional cnt,
    integer(i4), dimension(:), intent(in), optional stride,
    integer(i4), dimension(:), intent(in), optional map )
```

References check().

Here is the call graph for this function:

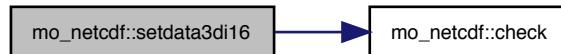


15.65.2.103 setdata3di16()

```
subroutine mo_netcdf::setdata3di16 (
    class(ncvariable), intent(in) self,
    integer(i2), dimension(:, :, :), intent(in) values,
    integer(i4), dimension(:), intent(in), optional start,
    integer(i4), dimension(:), intent(in), optional cnt,
    integer(i4), dimension(:), intent(in), optional stride,
    integer(i4), dimension(:), intent(in), optional map )
```

References check().

Here is the call graph for this function:



15.65.2.104 setdata3di32()

```
subroutine mo_netcdf::setdata3di32 (
    class(ncvariable), intent(in) self,
    integer(i4), dimension(:, :, :), intent(in) values,
    integer(i4), dimension(:), intent(in), optional start,
    integer(i4), dimension(:), intent(in), optional cnt,
    integer(i4), dimension(:), intent(in), optional stride,
    integer(i4), dimension(:), intent(in), optional map )
```

References check().

Here is the call graph for this function:

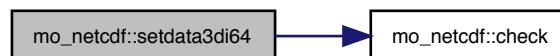


15.65.2.105 setdata3di64()

```
subroutine mo_ncdf::setdata3di64 (
    class(ncvariable), intent(in) self,
    integer(i8), dimension(:, :, :), intent(in) values,
    integer(i4), dimension(:), intent(in), optional start,
    integer(i4), dimension(:), intent(in), optional cnt,
    integer(i4), dimension(:), intent(in), optional stride,
    integer(i4), dimension(:), intent(in), optional map )
```

References check().

Here is the call graph for this function:



15.65.2.106 setdata3di8()

```
subroutine mo_ncdf::setdata3di8 (
    class(ncvariable), intent(in) self,
    integer(i1), dimension(:, :, :), intent(in) values,
    integer(i4), dimension(:), intent(in), optional start,
    integer(i4), dimension(:), intent(in), optional cnt,
    integer(i4), dimension(:), intent(in), optional stride,
    integer(i4), dimension(:), intent(in), optional map )
```

References check().

Here is the call graph for this function:



15.65.2.107 setdata4df32()

```
subroutine mo_ncdf::setdata4df32 (
    class(ncvariable), intent(in) self,
    real(sp), dimension(:, :, :, :), intent(in) values,
    integer(i4), dimension(:), intent(in), optional start,
    integer(i4), dimension(:), intent(in), optional cnt,
    integer(i4), dimension(:), intent(in), optional stride,
    integer(i4), dimension(:), intent(in), optional map )
```

References check().

Here is the call graph for this function:



15.65.2.108 setdata4df64()

```
subroutine mo_ncdf::setdata4df64 (
    class(ncvariable), intent(in) self,
    real(dp), dimension(:, :, :, :), intent(in) values,
    integer(i4), dimension(:), intent(in), optional start,
    integer(i4), dimension(:), intent(in), optional cnt,
    integer(i4), dimension(:), intent(in), optional stride,
    integer(i4), dimension(:), intent(in), optional map )
```

References check().

Here is the call graph for this function:

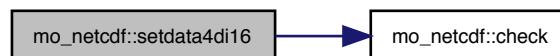


15.65.2.109 setdata4di16()

```
subroutine mo_ncdf::setdata4di16 (
    class(ncvariable), intent(in) self,
    integer(i2), dimension(:, :, :, :), intent(in) values,
    integer(i4), dimension(:), intent(in), optional start,
    integer(i4), dimension(:), intent(in), optional cnt,
    integer(i4), dimension(:), intent(in), optional stride,
    integer(i4), dimension(:), intent(in), optional map )
```

References check().

Here is the call graph for this function:

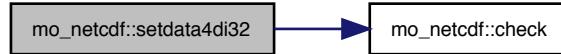


15.65.2.110 setdata4di32()

```
subroutine mo_ncdf::setdata4di32 (
    class(ncvariable), intent(in) self,
    integer(i4), dimension(:, :, :, :), intent(in) values,
    integer(i4), dimension(:), intent(in), optional start,
    integer(i4), dimension(:), intent(in), optional cnt,
    integer(i4), dimension(:), intent(in), optional stride,
    integer(i4), dimension(:), intent(in), optional map )
```

References check().

Here is the call graph for this function:

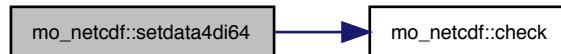


15.65.2.111 setdata4di64()

```
subroutine mo_ncdf::setdata4di64 (
    class(ncvariable), intent(in) self,
    integer(i8), dimension(:, :, :, :), intent(in) values,
    integer(i4), dimension(:), intent(in), optional start,
    integer(i4), dimension(:), intent(in), optional cnt,
    integer(i4), dimension(:), intent(in), optional stride,
    integer(i4), dimension(:), intent(in), optional map )
```

References check().

Here is the call graph for this function:



15.65.2.112 setdata4di8()

```
subroutine mo_ncdf::setdata4di8 (
    class(ncvariable), intent(in) self,
    integer(i1), dimension(:, :, :, :), intent(in) values,
    integer(i4), dimension(:), intent(in), optional start,
    integer(i4), dimension(:), intent(in), optional cnt,
    integer(i4), dimension(:), intent(in), optional stride,
    integer(i4), dimension(:), intent(in), optional map )
```

References check().

Here is the call graph for this function:



15.65.2.113 setdata5df32()

```
subroutine mo_ncdf::setdata5df32 (
    class(ncvariable), intent(in) self,
    real(sp), dimension(:, :, :, :, :), intent(in) values,
    integer(i4), dimension(:), intent(in), optional start,
    integer(i4), dimension(:), intent(in), optional cnt,
    integer(i4), dimension(:), intent(in), optional stride,
    integer(i4), dimension(:), intent(in), optional map )
```

References check().

Here is the call graph for this function:

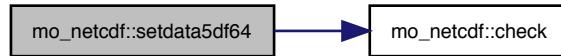


15.65.2.114 setdata5df64()

```
subroutine mo_ncdf::setdata5df64 (
    class(ncvariable), intent(in) self,
    real(dp), dimension(:, :, :, :, :), intent(in) values,
    integer(i4), dimension(:), intent(in), optional start,
    integer(i4), dimension(:), intent(in), optional cnt,
    integer(i4), dimension(:), intent(in), optional stride,
    integer(i4), dimension(:), intent(in), optional map )
```

References check().

Here is the call graph for this function:

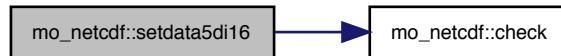


15.65.2.115 setdata5di16()

```
subroutine mo_ncdf::setdata5di16 (
    class(ncvariable), intent(in) self,
    integer(i2), dimension(:, :, :, :, :), intent(in) values,
    integer(i4), dimension(:), intent(in), optional start,
    integer(i4), dimension(:), intent(in), optional cnt,
    integer(i4), dimension(:), intent(in), optional stride,
    integer(i4), dimension(:), intent(in), optional map )
```

References check().

Here is the call graph for this function:

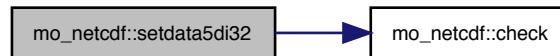


15.65.2.116 setdata5di32()

```
subroutine mo_ncdf::setdata5di32 (
    class(ncvariable), intent(in) self,
    integer(i4), dimension(:, :, :, :, :), intent(in) values,
    integer(i4), dimension(:), intent(in), optional start,
    integer(i4), dimension(:), intent(in), optional cnt,
    integer(i4), dimension(:), intent(in), optional stride,
    integer(i4), dimension(:), intent(in), optional map )
```

References check().

Here is the call graph for this function:

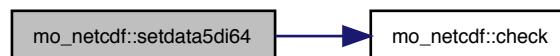


15.65.2.117 setdata5di64()

```
subroutine mo_ncdf::setdata5di64 (
    class(ncvariable), intent(in) self,
    integer(i8), dimension(:, :, :, :, :), intent(in) values,
    integer(i4), dimension(:), intent(in), optional start,
    integer(i4), dimension(:), intent(in), optional cnt,
    integer(i4), dimension(:), intent(in), optional stride,
    integer(i4), dimension(:), intent(in), optional map )
```

References check().

Here is the call graph for this function:

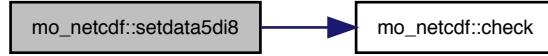


15.65.2.118 setdata5di8()

```
subroutine mo_ncdf::setdata5di8 (
    class(ncvariable), intent(in) self,
    integer(i1), dimension(:, :, :, :, :), intent(in) values,
    integer(i4), dimension(:), intent(in), optional start,
    integer(i4), dimension(:), intent(in), optional cnt,
    integer(i4), dimension(:), intent(in), optional stride,
    integer(i4), dimension(:), intent(in), optional map )
```

References check().

Here is the call graph for this function:



15.65.2.119 setdatascalarf32()

```
subroutine mo_ncdf::setdatascalarf32 (
    class(ncvariable), intent(in) self,
    real(sp), intent(in) values,
    integer(i4), dimension(:), intent(in), optional start )
```

References check().

Here is the call graph for this function:



15.65.2.120 setdatascalarf64()

```
subroutine mo_ncdf::setdatascalarf64 (
    class(ncvariable), intent(in) self,
    real(dp), intent(in) values,
    integer(i4), dimension(:), intent(in), optional start )
```

References check().

Here is the call graph for this function:



15.65.2.121 setdatascalari16()

```
subroutine mo_ncdf::setdatascalari16 (
    class(ncvariable), intent(in) self,
    integer(i2), intent(in) values,
    integer(i4), dimension(:), intent(in), optional start )
```

References check().

Here is the call graph for this function:

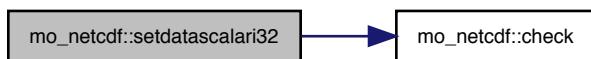


15.65.2.122 setdatascalari32()

```
subroutine mo_ncdf::setdatascalari32 (
    class(ncvariable), intent(in) self,
    integer(i4), intent(in) values,
    integer(i4), dimension(:), intent(in), optional start )
```

References check().

Here is the call graph for this function:



15.65.2.123 setdatascalari64()

```
subroutine mo_ncdf::setdatascalari64 (
    class(ncvariable), intent(in) self,
    integer(i8), intent(in) values,
    integer(i4), dimension(:), intent(in), optional start )
```

References check().

Here is the call graph for this function:



15.65.2.124 setdatascalari8()

```
subroutine mo_ncdf::setdatascalari8 (
    class(ncvariable), intent(in) self,
    integer(il), intent(in) values,
    integer(i4), dimension(:), intent(in), optional start )
```

References check().

Here is the call graph for this function:



15.65.2.125 setdimension()

```
type(ncdimension) function mo_ncdf::setdimension (
    class(ncdataset), intent(in) self,
    character(*), intent(in) name,
    integer(i4), intent(in) length )
```

References check().

Here is the call graph for this function:



15.65.2.126 setglobalattributechar()

```
subroutine mo_ncdf::setglobalattributechar (
    class(ncdataset), intent(in) self,
    character(*), intent(in) name,
    character(*), intent(in) data )
```

References check().

Here is the call graph for this function:

**15.65.2.127 setglobalattributef32()**

```
subroutine mo_ncdf::setglobalattributef32 (
    class(ncdataset), intent(in) self,
    character(*), intent(in) name,
    real(sp), intent(in) data )
```

References check().

Here is the call graph for this function:

**15.65.2.128 setglobalattributef64()**

```
subroutine mo_ncdf::setglobalattributef64 (
    class(ncdataset), intent(in) self,
    character(*), intent(in) name,
    real(dp), intent(in) data )
```

References check().

Here is the call graph for this function:



15.65.2.129 setglobalattributei16()

```
subroutine mo_ncdf::setglobalattributei16 (
    class(ncdataset), intent(in) self,
    character(*), intent(in) name,
    integer(i2), intent(in) data )
```

References check().

Here is the call graph for this function:



15.65.2.130 setglobalattributei32()

```
subroutine mo_ncdf::setglobalattributei32 (
    class(ncdataset), intent(in) self,
    character(*), intent(in) name,
    integer(i4), intent(in) data )
```

References check().

Here is the call graph for this function:



15.65.2.131 setglobalattributei64()

```
subroutine mo_ncdf::setglobalattributei64 (
    class(ncdataset), intent(in) self,
    character(*), intent(in) name,
    integer(i8), intent(in) data )
```

References check().

Here is the call graph for this function:



15.65.2.132 setglobalattributei8()

```
subroutine mo_ncdf::setglobalattributei8 (
    class(ncdataset), intent(in) self,
    character(*), intent(in) name,
    integer(i1), intent(in) data )
```

References check().

Here is the call graph for this function:



15.65.2.133 setvariableattributechar()

```
subroutine mo_ncdf::setvariableattributechar (
    class(ncvariable), intent(in) self,
    character(*), intent(in) name,
    character(*), intent(in) data )
```

References check().

Here is the call graph for this function:



15.65.2.134 setvariableattribute32()

```
subroutine mo_ncdf::setvariableattribute32 (
    class(ncvariable), intent(in) self,
    character(*), intent(in) name,
    real(sp), intent(in) data )
```

References check().

Here is the call graph for this function:



15.65.2.135 setvariableattribute64()

```
subroutine mo_ncdf::setvariableattribute64 (
    class(ncvariable), intent(in) self,
    character(*), intent(in) name,
    real(dp), intent(in) data )
```

References check().

Here is the call graph for this function:



15.65.2.136 setvariableattributei16()

```
subroutine mo_ncdf::setvariableattributei16 (
    class(ncvariable), intent(in) self,
    character(*), intent(in) name,
    integer(i2), intent(in) data )
```

References check().

Here is the call graph for this function:

**15.65.2.137 setvariableattributei32()**

```
subroutine mo_ncdf::setvariableattributei32 (
    class(ncvariable), intent(in) self,
    character(*), intent(in) name,
    integer(i4), intent(in) data )
```

References check().

Here is the call graph for this function:

**15.65.2.138 setvariableattributei64()**

```
subroutine mo_ncdf::setvariableattributei64 (
    class(ncvariable), intent(in) self,
    character(*), intent(in) name,
    integer(i8), intent(in) data )
```

References check().

Here is the call graph for this function:



15.65.2.139 setvariableattributei8()

```
subroutine mo_ncdf::setvariableattributei8 (
    class(ncvariable), intent(in) self,
    character(*), intent(in) name,
    integer(il), intent(in) data )
```

References check().

Here is the call graph for this function:



15.65.2.140 setvariablefillvaluef32()

```
subroutine mo_ncdf::setvariablefillvaluef32 (
    class(ncvariable), intent(in) self,
    real(sp), intent(in) fvalue )
```

15.65.2.141 setvariablefillvaluef64()

```
subroutine mo_ncdf::setvariablefillvaluef64 (
    class(ncvariable), intent(in) self,
    real(dp), intent(in) fvalue )
```

15.65.2.142 setvariablefillvaluei16()

```
subroutine mo_ncdf::setvariablefillvaluei16 (
```

```
class(ncvariable), intent(in) self,
integer(i2), intent(in) fvalue )
```

15.65.2.143 setvariablefillvaluei32()

```
subroutine mo_ncdf::setvariablefillvaluei32 (
    class(ncvariable), intent(in) self,
    integer(i4), intent(in) fvalue )
```

15.65.2.144 setvariablefillvaluei64()

```
subroutine mo_ncdf::setvariablefillvaluei64 (
    class(ncvariable), intent(in) self,
    integer(i8), intent(in) fvalue )
```

15.65.2.145 setvariablefillvaluei8()

```
subroutine mo_ncdf::setvariablefillvaluei8 (
    class(ncvariable), intent(in) self,
    integer(i1), intent(in) fvalue )
```

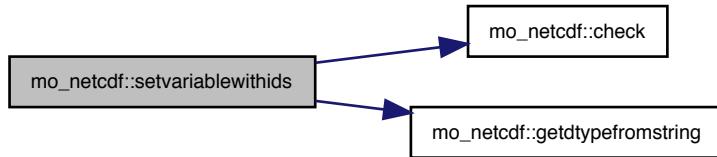
15.65.2.146 setvariablewithids()

```
type(ncvariable) function mo_ncdf::setvariablewithids (
    class(ncdataset), intent(in) self,
    character(*), intent(in) name,
    character(3), intent(in) dtype,
    integer(i4), dimension(:), intent(in) dimensions,
    logical, intent(in), optional contiguous,
    integer(i4), dimension(:), intent(in), optional chunksizes,
    integer(i4), intent(in), optional deflate_level,
    logical, intent(in), optional shuffle,
    logical, intent(in), optional fletcher32,
    integer(i4), intent(in), optional endianness,
    integer(i4), intent(in), optional cache_size,
    integer(i4), intent(in), optional cache_nelems,
    integer(i4), intent(in), optional cache_preemption )
```

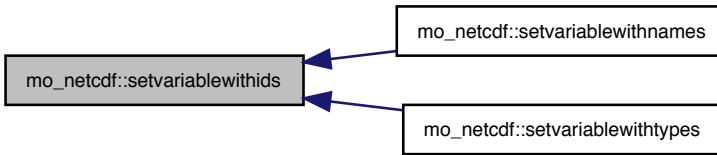
References `check()`, and `getdtypefromstring()`.

Referenced by `setvariablewithnames()`, and `setvariablewithtypes()`.

Here is the call graph for this function:



Here is the caller graph for this function:



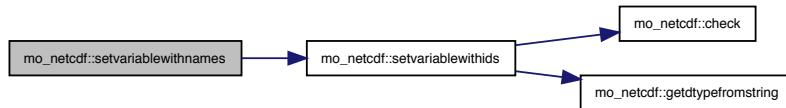
15.65.2.147 setvariablewithnames()

```

type(ncvariable) function mo_netcdf::setvariablewithnames (
    class(ncdataset), intent(in) self,
    character(*), intent(in) name,
    character(3), intent(in) dtype,
    character(*), dimension(:), intent(in) dimensions,
    logical, intent(in), optional contiguous,
    integer(i4), dimension(:), intent(in), optional chunksizes,
    integer(i4), intent(in), optional deflate_level,
    logical, intent(in), optional shuffle,
    logical, intent(in), optional fletcher32,
    integer(i4), intent(in), optional endianness,
    integer(i4), intent(in), optional cache_size,
    integer(i4), intent(in), optional cache_nelems,
    integer(i4), intent(in), optional cache_premption )
  
```

References setvariablewithids().

Here is the call graph for this function:



15.65.2.148 setvariablewithtypes()

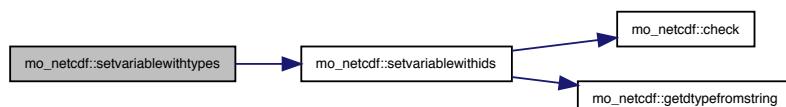
```

type(ncvariable) function mo_netcdf::setvariablewithtypes (
    class(ncdataset), intent(in) self,
    character(*), intent(in) name,
    character(3), intent(in) dtype,
    type(ncdimension), dimension(:), intent(in) dimensions,
    logical, intent(in), optional contiguous,
    integer(i4), dimension(:), intent(in), optional chunksizes,
    integer(i4), intent(in), optional deflate_level,
    logical, intent(in), optional shuffle,
    logical, intent(in), optional fletcher32,
    integer(i4), intent(in), optional endianness,
    integer(i4), intent(in), optional cache_size,
    integer(i4), intent(in), optional cache_nelems,
    integer(i4), intent(in), optional cache_premption )

```

References setvariablewithids().

Here is the call graph for this function:



15.66 mo_neutrons Module Reference

Models to predict neutron intensities above soils.

Functions/Subroutines

- subroutine, public **desiletsn0** (SoilMoisture, Horizons, N0, neutrons)
Calculate neutrons from soil moisture in the first layer.
- subroutine, public **cosmic** (SoilMoisture, Horizons, params, neutron_integral_AFast, neutrons)
Calculate neutrons from soil moisture in all layers.

- subroutine `oldintegration` (res, c)
TODO: add description.
- subroutine, public `tabularintegralfast` (integral, maxC)
Save approximation data for A_fast.
- subroutine `approx_mon_int` (res, f, c, xmin, xmax, eps, steps, fxmin, fxmax)
TODO: add description.
- recursive subroutine `approx_mon_int_steps` (res, f, c, xmin, xmax, eps, steps, fxmin, fxmax)
TODO: add description.
- recursive subroutine `approx_mon_int_eps` (res, f, c, xmin, xmax, eps, fxmin, fxmax)
TODO: add description.
- subroutine `lookupintegral` (res, integral, c)
TODO: add description.
- real(dp) function `intgrandfast` (c, phi)
TODO: add description.

15.66.1 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.

15.66.2 Function/Subroutine Documentation

15.66.2.1 `approx_mon_int()`

```
subroutine mo_neutrons::approx_mon_int (
    real(dp) res,
    real(dp), external f,
    real(dp), intent(in) c,
    real(dp), intent(in) xmin,
    real(dp), intent(in) xmax,
    real(dp), intent(in), optional eps,
    integer(i4), intent(in), optional steps,
    real(dp), intent(in), optional fxmin,
    real(dp), intent(in), optional fxmax )
```

TODO: add description.

TODO: add description

Parameters

in	<code>real(dp) :: c</code>		
in	<code>real(dp) :: xmin</code>		

Parameters

in	<i>real(dp) :: xmax</i>	
in	<i>real(dp), optional :: eps</i>	
in	<i>integer(i4), optional :: steps</i>	
in	<i>real(dp), optional :: fxmin</i>	
in	<i>real(dp), optional :: fxmax</i>	

Authors

Robert Schweißpe

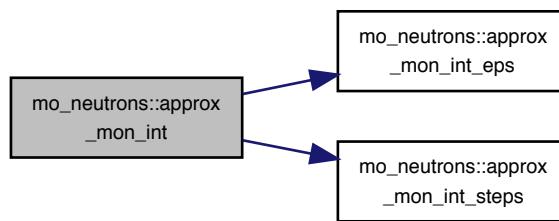
Date

Jun 2018

References approx_mon_int_eps(), and approx_mon_int_steps().

Referenced by tabularintegralafast().

Here is the call graph for this function:



Here is the caller graph for this function:



15.66.2.2 approx_mon_int_eps()

```

recursive subroutine mo_neutrons::approx_mon_int_eps (
    real(dp) res,
    real(dp), external f,
    real(dp), intent(in) c,
    real(dp), intent(in) xmin,
    real(dp), intent(in) xmax,
  
```

```

    real(dp), intent(in)  eps,
    real(dp), intent(in)  fxmin,
    real(dp), intent(in)  fxmax )  [private]

```

TODO: add description.

TODO: add description

Parameters

in	<i>real(dp) :: c</i>	
in	<i>real(dp) :: xmin</i>	
in	<i>real(dp) :: xmax</i>	
in	<i>real(dp) :: eps</i>	
in	<i>real(dp) :: fxmin</i>	
in	<i>real(dp) :: fxmax</i>	

Authors

Robert Schwepppe

Date

Jun 2018

Referenced by `approx_mon_int()`.

Here is the caller graph for this function:



15.66.2.3 approx_mon_int_steps()

```

recursive subroutine mo_neurons::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	<i>real(dp) :: c</i>	
in	<i>real(dp) :: xmin</i>	

Parameters

in	<i>real(dp)</i> :: <i>xmax</i>	
in	<i>real(dp)</i> :: <i>eps</i>	
in	<i>integer(i4)</i> :: <i>steps</i>	
in	<i>real(dp)</i> :: <i>fxmin</i>	
in	<i>real(dp)</i> :: <i>fxmax</i>	

Authors

Robert Scheweppe

Date

Jun 2018

Referenced by `approx_mon_int()`.

Here is the caller graph for this function:

15.66.2.4 `cosmic()`

```

subroutine, public mo_neutrons::cosmic (
    real(dp), dimension(:), intent(in) SoilMoisture,
    real(dp), dimension(:), intent(in) Horizons,
    real(dp), dimension(:), intent(in) params,
    real(dp), dimension(:), intent(in) neutron_integral_AFast,
    real(dp), intent(inout) neutrons )
  
```

Calculate neutrons from soil moisture in all layers.

Neutron counts above the ground (one value per cell in mHM) can be derived by a simplified physical neutron transport simulation. Fast cosmic-Ray neutrons are generated in the soil and attenuated differently in water and soil. The remaining neutrons that reached the surface relate to the profile of soil water content below. Variables like N, alpha and L3 are site-specific and need to be calibrated. ADDITIONAL INFORMATION COSMIC model based on Shuttleworth et al. 2013 Horizons(:) must not be zero. see supplementaries in literature J. Shuttleworth, R. Rosolem, M. Zreda, and T. Franz, The COsmic-ray Soil Moisture Interaction Code (COSMIC) for use in data assimilation, HESS, 17, 3205-3217, 2013, doi:10.5194/hess-17-3205-2013 Support and Code: <http://cosmos.hwr.arizona.edu/Software/cosmic.html>

Parameters

in	<i>real(dp), dimension(:)</i> :: <i>SoilMoisture</i>	Soil Moisture
in	<i>real(dp), dimension(:)</i> :: <i>Horizons</i>	Horizon depths
in	<i>real(dp), dimension(:)</i> :: <i>params</i>	! N0, N1, N2, alpha0, alpha1, L30, L31
in	<i>real(dp), dimension(:)</i> :: <i>neutron_integral_AFast</i>	Tabular for Int Approx
in, out	<i>real(dp)</i> :: <i>neutrons</i>	Neutron counts

Authors

Martin Schroen, originally written by Rafael Rosolem

Date

Mar 2015

References `mo_mhm_constants::cosmic_alpha`, `mo_mhm_constants::cosmic_bd`, `mo_mhm_constants::cosmic_l1`, `mo_mhm_constants::cosmic_l2`, `mo_mhm_constants::cosmic_l3`, `mo_mhm_constants::cosmic_l4`, `mo_mhm_constants::cosmic_n`, `mo_mhm_constants::cosmic_vwclat`, `mo_mhm_constants::h2odens`, `lookupintegral()`, and `mo_constants::pi_dp`.

Referenced by `mo_mhm::mhm()`.

Here is the call graph for this function:



Here is the caller graph for this function:



15.66.2.5 desiletsn0()

```

subroutine, public mo_neutrons::desiletsn0 (
    real(dp), dimension(:), intent(in) SoilMoisture,
    real(dp), dimension(:), intent(in) Horizons,
    real(dp), intent(in) N0,
    real(dp), intent(inout) neutrons )

```

Calculate neutrons from soil moisture in the first layer.

Using the N0-relation derived by Desilets, neutron counts above the ground (one value per cell in mHM) can be derived by a semi-empirical, semi-physical relation. The result depends on N0, the neutron counts for 0% soil moisture. 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:10.1029/2009WR008726.

Parameters

in	<i>real(dp)</i> , <i>dimension(:)</i> :: <i>SoilMoisture</i>	Soil Moisture
in	<i>real(dp)</i> , <i>dimension(:)</i> :: <i>Horizons</i>	Horizon depths
in	<i>real(dp)</i> :: <i>N0</i>	dry neutron counts
in, out	<i>real(dp)</i> :: <i>neutrons</i>	Neutron counts

Authors

Martin Schroen

Date

Mar 2015

References `mo_mhm_constants::desilets_a0`, `mo_mhm_constants::desilets_a1`, and `mo_mhm_constants::desilets_a2`.

Referenced by `mo_mhm::mhm()`.

Here is the caller graph for this function:

15.66.2.6 `intgrandfast()`

```
real(dp) function mo_neutrons::intgrandfast (
    real(dp), intent(in) c,
    real(dp), intent(in) phi )
```

TODO: add description.

TODO: add description

Parameters

in	<i>real(dp)</i> :: <i>c</i>	
in	<i>real(dp)</i> :: <i>phi</i>	

Authors

Robert Scheweppe

Date

Jun 2018

Referenced by `tabularintegralafast()`.

Here is the caller graph for this function:



15.66.2.7 lookupintegral()

```
subroutine mo_neutrons::lookupintegral (
    real(dp) res,
    real(dp), dimension(:), intent(in) integral,
    real(dp), intent(in) c ) [private]
```

TODO: add description.

TODO: add description

Parameters

in	real(dp), dimension(:) :: integral	
in	real(dp) :: c	

Authors

Robert Schweißpape

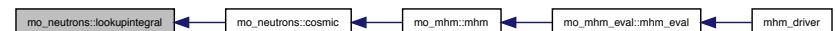
Date

Jun 2018

References mo_kind::i4, and mo_constants::pi_dp.

Referenced by cosmic().

Here is the caller graph for this function:



15.66.2.8 oldintegration()

```
subroutine mo_neutrons::oldintegration (
    real(dp) res,
    real(dp), intent(in) c )
```

TODO: add description.

TODO: add description

Parameters

in	real(dp) :: c	
----	---------------	--

Authors

Robert Schweiß

Date

Jun 2018

References mo_constants::pi_dp.

15.66.2.9 tabularintegralafast()

```
subroutine, public mo_neutrons::tabularintegralafast (
    real(dp), dimension(:) integral,
    real(dp), intent(in) maxC )
```

Save approximation data for A_fast.

The COSMIC subroutine needs A_fast to be calculated. $A_{\text{fast}} = \int_{-c}^0 \exp(-\Lambda_{\text{fast}}(z)/\cos(\phi)) \, dz$. This subroutine stores data for intsize values for $c = \Lambda_{\text{fast}}(z)$ 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 $0 < c < \text{maxC}$, otherwise the recursive approximation is used. TabularIntegralAFast: a tabular for calculations with splines intsize and maxC must be positive intsize=8000, maxC=20.0_dp see splines for example

Parameters

in	real(dp) :: maxC	! maximum value for A_fast
----	------------------	----------------------------

Authors

Maren Kaluza

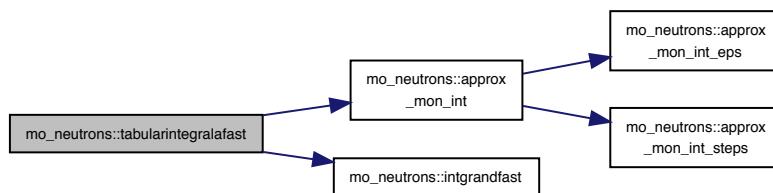
Date

Nov 2017

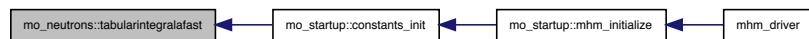
References `approx_mon_int()`, `intgrandfast()`, and `mo_constants::pi_dp`.

Referenced by `mo_startup::constants_init()`.

Here is the call graph for this function:



Here is the caller graph for this function:



15.67 mo_nml Module Reference

Deal with namelist files.

Functions/Subroutines

- subroutine, public `open_nml` (file, unit, quiet)
Open a namelist file.
- subroutine, public `close_nml` (unit)
Close a namelist file.
- subroutine, public `position_nml` (name, unit, status, first)
Position a namelist file.

Variables

- integer(i4), parameter, public `positioned` = 0
Information: file pointer set to namelist group.
- integer(i4), parameter, public `missing` = 1
Error: namelist group is missing.
- integer(i4), parameter, public `length_error` = 2
Error: namelist group name too long.
- integer(i4), parameter, public `read_error` = 3
Error occurred during read of namelist file.
- integer, save, public `nunitnml` = -1

15.67.1 Detailed Description

Deal with namelist files.

This module provides routines to open, close and position namelist files.

Authors

Matthias Cuntz

Date

Jan 2011

15.67.2 Function/Subroutine Documentation

15.67.2.1 close_nml()

```
subroutine, public mo_nml::close_nml (
    integer, intent(in), optional unit )
```

Close a namelist file.

Parameters

in	<i>integer, optional :: unit</i>	namelist unit
----	----------------------------------	---------------

Author

Matthias Cuntz - modified from Echam5, (C) MPI-MET, Hamburg, Germany

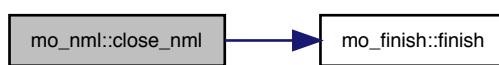
Date

Dec 2011

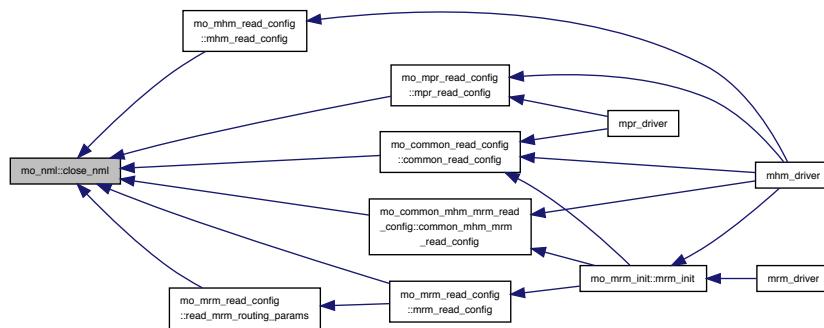
References mo_finish::finish(), and nunitnml.

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

Here is the call graph for this function:



Here is the caller graph for this function:



15.67.2.2 open_nml()

```
subroutine, public mo_nml::open_nml (
    character(len = *), intent(in) file,
    integer, intent(in) unit,
    logical, intent(in), optional quiet )
```

Open a namelist file.

Parameters

in	<i>character(len=*) :: file</i>	namelist filename
in	<i>integer :: unit</i>	namelist unit
in	<i>logical, optional :: quiet</i>	Be verbose or not (default: .true.) .true.: no messages .false.: write out messages

Author

Matthias Cuntz - modified from Echam5, (C) MPI-MET, Hamburg, Germany

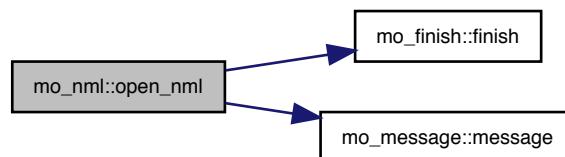
Date

Dec 2011

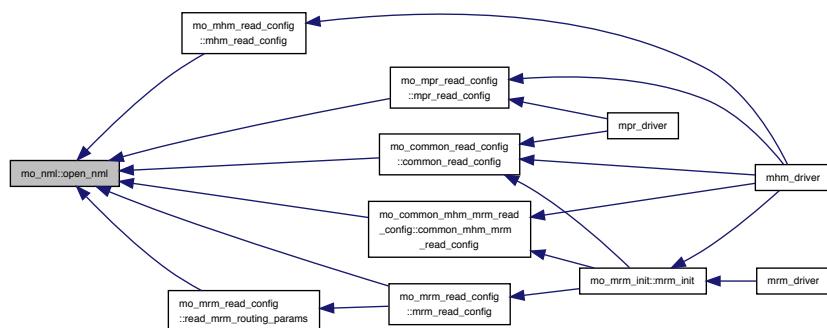
References mo_finish::finish(), mo_message::message(), mo_message::message_text, and nunitnml.

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

Here is the call graph for this function:



Here is the caller graph for this function:



15.67.2.3 position_nml()

```

subroutine, public mo_nml::position_nml (
    character(len = *), intent(in)  name,
    integer, intent(in), optional unit,
    integer(i4), intent(out), optional status,
    logical, intent(in), optional first )
  
```

Position a namelist 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	<i>character(len=*) :: name</i>	namelist name (case independent)
in	<i>integer, optional :: unit</i>	namelist unit (default: nunitnml)
in	<i>logical, optional :: first</i>	start search at beginning, i.e. rewind the namelist first (default: .true.) .true.: rewind .false.: continue search from current file pointer
out	<i>integer(i4), optional :: status</i>	Set on output to either of POSITIONED (0) - correct MISSING (1) - name not found LENGTH_ERROR (2) - namelist length longer then 256 characters READ_ERROR (3) - error while reading namelist file

Author

Matthias Cuntz - modified from Echam5, (C) MPI-MET, Hamburg, Germany

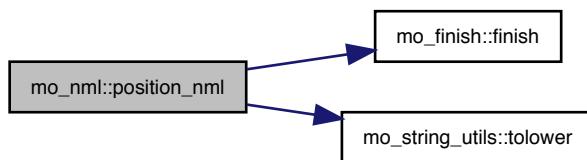
Date

Dec 2011

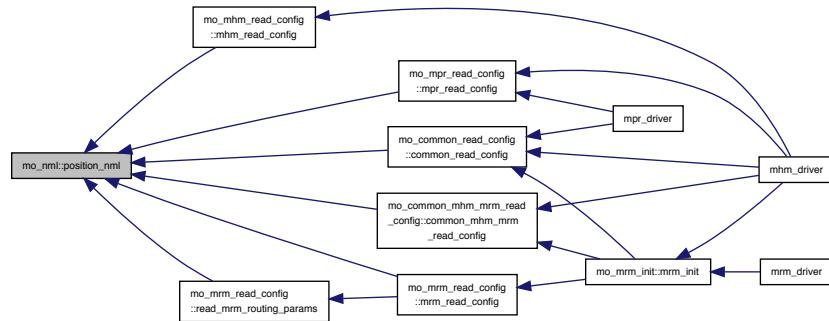
References `mo_finish::finish()`, `length_error`, `mo_message::message_text`, `missing`, `nunitnml`, `positioned`, `read_error`, and `mo_string_utils::tolower()`.

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

Here is the call graph for this function:



Here is the caller graph for this function:



15.67.3 Variable Documentation

15.67.3.1 length_error

```
integer(i4), parameter, public mo_nml::length_error = 2
```

Error: namelist group name too long.

Referenced by position_nml().

15.67.3.2 missing

```
integer(i4), parameter, public mo_nml::missing = 1
```

Error: namelist group is missing.

Referenced by position_nml().

15.67.3.3 nunitnml

```
integer, save, public mo_nml::nunitnml = -1
```

Referenced by close_nml(), open_nml(), and position_nml().

15.67.3.4 positioned

```
integer(i4), parameter, public mo_nml::positioned = 0
```

Information: file pointer set to namelist group.

Referenced by position_nml().

15.67.3.5 read_error

```
integer(i4), parameter, public mo_nml::read_error = 3
```

Error occurred during read of namelist file.

Referenced by position_nml().

15.68 mo_objective_function Module Reference

Objective Functions for Optimization of mHM.

Functions/Subroutines

- real(dp) function, public [objective](#) (parameterset, eval, arg1, arg2, arg3)
Wrapper for objective functions.
- real(dp) function [objective_sm_kge_catchment_avg](#) (parameterset, eval)
Objective function for soil moisture.
- real(dp) function [objective_sm_corr](#) (parameterset, eval)
Objective function for soil moisture.
- real(dp) function [objective_sm_pd](#) (parameterset, eval)
Objective function for soil moisture.
- real(dp) function [objective_sm_sse_standard_score](#) (parameterset, eval)
Objective function for soil moisture.
- real(dp) function [objective_kge_q_rmse_tws](#) (parameterset, eval)
Objective function of KGE for runoff and RMSE for basin_avg TWS (standarized scores)
- real(dp) function [objective_neutrons_kge_catchment_avg](#) (parameterset, eval)
Objective function for neutrons.
- real(dp) function [objective_et_kge_catchment_avg](#) (parameterset, eval)
Objective function for evapotranspiration (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

15.68.1 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

15.68.2 Function/Subroutine Documentation

15.68.2.1 extract_basin_avg_tws()

```
subroutine mo_objective_function::extract_basin_avg_tws (
    integer(i4), intent(in) basinId,
    real(dp), dimension(:, :), intent(in) tws,
    real(dp), dimension(:), intent(out), allocatable tws_sim,
    real(dp), dimension(:), intent(out), allocatable tws_obs,
    logical, dimension(:, ), intent(out), allocatable tws_obs_mask )
```

extracts basin average tws data from global variables

extracts simulated and measured basin average tws from global variables, such that they overlay exactly. For measured tws, only the tws during the evaluation period are cut, not succeeding nodata values. For simulated tws, warming days as well as succeeding nodata values are neglected. see use in this module above

Parameters

in	<i>integer(i4) :: basinId</i>	current basin Id
in	<i>real(dp), dimension(:, :) :: tws</i>	simulated basin average tws
out	<i>real(dp), dimension(:) :: tws_sim</i>	aggregated simulated
out	<i>real(dp), dimension(:) :: tws_obs</i>	extracted measured
out	<i>logical, dimension(:,) :: tws_obs_mask</i>	mask of no data values

Authors

Stephan Thober

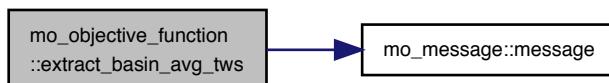
Date

Oct 2015

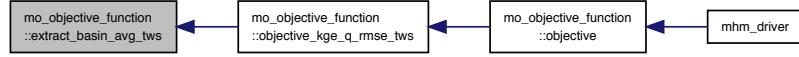
References `mo_global_variables::basin_avg_tws_obs`, `mo_common_constants::eps_dp`, `mo_common_mhm_mrm_variables::evalper`, `mo_message::message()`, `mo_global_variables::nmeasperday_tws`, `mo_common_constants::nodata_dp`, `mo_common_mhm_mrm_variables::ntstepday`, and `mo_common_mhm_mrm_variables::warmingdays`.

Referenced by `objective_kge_q_rmse_tws()`.

Here is the call graph for this function:



Here is the caller graph for this function:



15.68.2.2 `objective()`

```

real(dp) function, public mo_objective_function::objective (
    real(dp), dimension(:), intent(in) parameterset,
    procedure(eval_interface), intent(in), pointer eval,
    real(dp), intent(in), optional arg1,
    real(dp), intent(out), optional arg2,
    real(dp), intent(out), optional arg3 )
  
```

Wrapper for objective functions.

The functions selects the objective function case defined in a namelist, i.e. the global variable `opti_function`. It return the objective function value for a specific parameter set.

Parameters

in	<code>REAL(dp), DIMENSION(:) :: parameterset</code>	
in	<code>procedure(eval_interface) :: eval</code>	
in	<code>real(dp), optional :: arg1</code>	
out	<code>real(dp), optional :: arg2</code>	
out	<code>real(dp), optional :: arg3</code>	

Returns

real(dp) :: objective — objective function value (which will be e.g. minimized by an optimization routine like DDS)

Authors

Juliane Mai

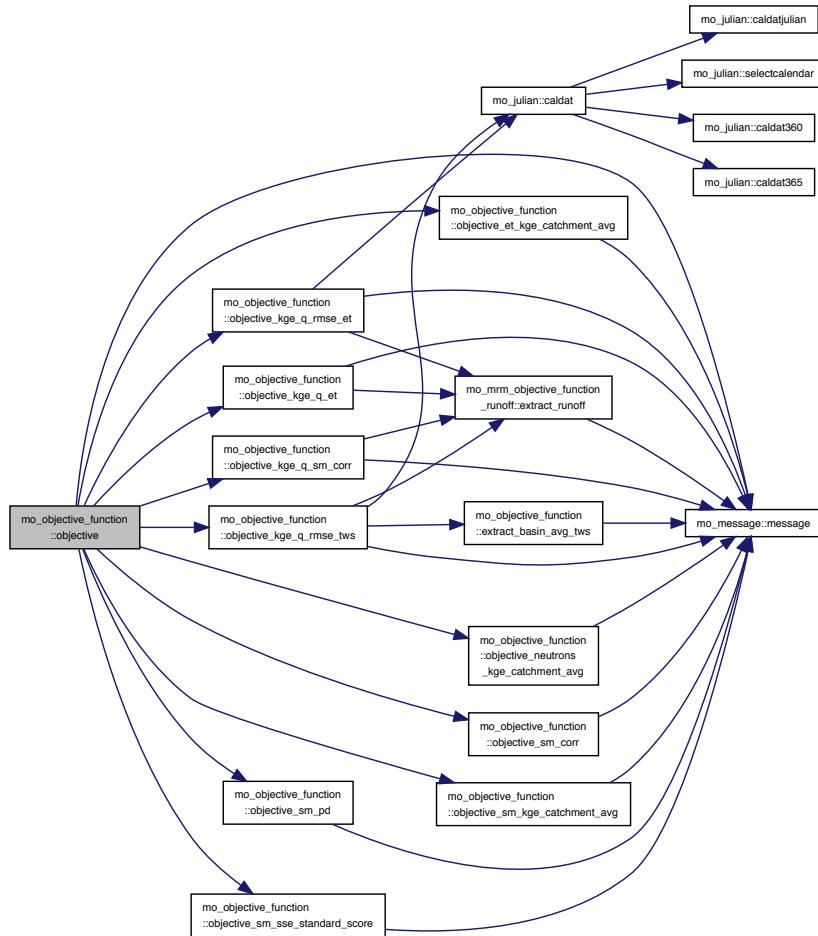
Date

Dec 2012

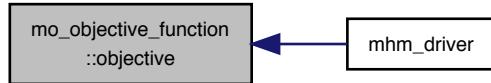
References `mo_message::message()`, `mo_common_constants::nodata_dp`, `objective_et_kge_catchment_avg()`, `objective_kge_q_et()`, `objective_kge_q_rmse_et()`, `objective_kge_q_rmse_tws()`, `objective_kge_q_sm_corr()`, `objective_neutrons_kge_catchment_avg()`, `objective_sm_corr()`, `objective_sm_kge_catchment_avg()`, `objective_sm_pd()`, `objective_sm_sse_standard_score()`, and `mo_common_mhm_mrm_variables::opti_function`.

Referenced by `mhm_driver()`.

Here is the call graph for this function:



Here is the caller graph for this function:



15.68.2.3 objective_et_kge_catchment_avg()

```
real(dp) function mo_objective_function::objective_et_kge_catchment_avg (
    real(dp), dimension(:), intent(in) parameterset,
    procedure(eval_interface), intent(in), pointer eval )
```

Objective function for evapotranspiration (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 simulated standard deviation to observed standard deviation is calculated and the objective function for a given basin i is

$$\phi_i = 1.0 - KGE_i$$

ϕ_i is the objective since we always apply minimization methods. The minimal value of ϕ_i is 0 for the optimal KGE of 1.0. Finally, the overall objective function value OF is estimated based on the power-6 norm to combine the ϕ_i from all basins N .

$$OF = \sqrt[6]{\sum ((1.0 - KGE_i)/N)^6}.$$

The observed data L1_et, L1_et_mask are global in this module.

Parameters

in	real(dp), dimension(:) :: parameterset	
in	procedure(eval_interface) :: eval	

Returns

real(dp) :: objective_et_kge_catchment_avg — objective function value (which will be e.g. minimized by an optimization routine)

Authors

Johannes Brenner

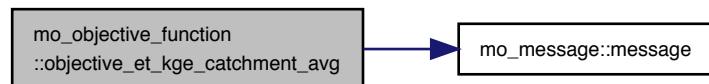
Date

Feb 2017

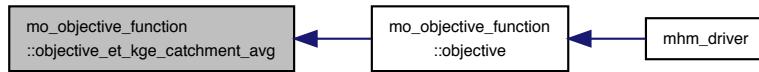
References mo_global_variables::l1_et, mo_global_variables::l1_et_mask, mo_common_variables::level1, mo_message::message(), mo_common_variables::nbasins, and mo_common_constants::nodata_dp.

Referenced by objective().

Here is the call graph for this function:



Here is the caller graph for this function:



15.68.2.4 objective_kge_q_et()

```

real(dp) function mo_objective_function::objective_kge_q_et (
    real(dp), dimension(:), intent(in) parameterset,
    procedure(eval_interface), intent(in), pointer eval )

```

Objective function of KGE for runoff and KGE for ET.

Objective function of KGE for runoff and KGE for ET. Further details can be found in the documentation of objective functions '14 - objective_multiple_gauges_kge_power6'.

Parameters

in	real(dp), dimension(:) :: parameterset	
in	procedure(eval_interface) :: eval	

Returns

real(dp) :: objective_kge_q_et — objective function value (which will be e.g. minimized by an optimization routine like DDS)

Authors

Johannes Brenner

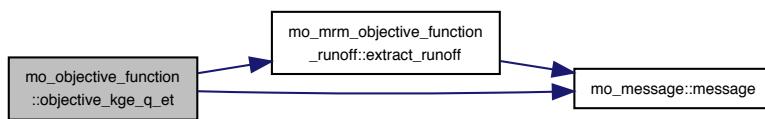
Date

July 2017

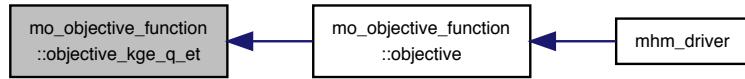
References `mo_mrm_objective_function_runoff::extract_runoff()`, `mo_global_variables::l1_et`, `mo_global_variables::l1_et_mask`, `mo_common_variables::level1`, `mo_message::message()`, and `mo_common_variables::nbasins`.

Referenced by `objective()`.

Here is the call graph for this function:



Here is the caller graph for this function:



15.68.2.5 `objective_kge_q_rmse_et()`

```
real(dp) function mo_objective_function::objective_kge_q_rmse_et (
    real(dp), dimension(:), intent(in) parameterset,
    procedure(eval_interface), intent(in), pointer eval )
```

Objective function of KGE for runoff and RMSE for basin_avg ET (standarized scores)

Objective function of KGE for runoff and RMSE for basin_avg ET (standarized scores)

Parameters

in	<code>real(dp), dimension(:) :: parameterset</code>	
in	<code>procedure(eval_interface) :: eval</code>	

Returns

`real(dp) :: objective_kge_q_rmse_et` — objective function value (which will be e.g. minimized by an optimization routine like DDS)

Authors

Johannes Brenner

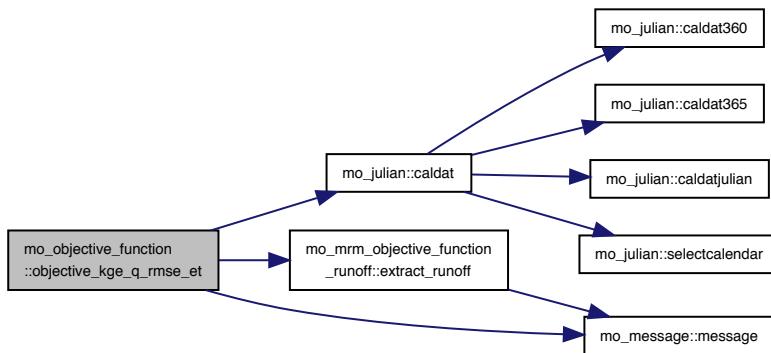
Date

July 2017

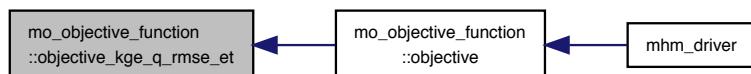
References `mo_julian::caldat()`, `mo_common_constants::eps_dp`, `mo_common_mhm_mrm_variables::evalper`, `mo_mrm_objective_function_runoff::extract_runoff()`, `mo_global_variables::l1_et`, `mo_global_variables::l1_et ← mask`, `mo_common_variables::level1`, `mo_message::message()`, `mo_common_variables::nbasins`, `mo_common_constants::nodata_dp`, and `mo_global_variables::timestep_et_input`.

Referenced by `objective()`.

Here is the call graph for this function:



Here is the caller graph for this function:

15.68.2.6 `objective_kge_q_rmse_tws()`

```
real(dp) function mo_objective_function::objective_kge_q_rmse_tws (
    real(dp), dimension(:), intent(in) parameterset,
    procedure(eval_interface), intent(in), pointer eval )
```

Objective function of KGE for runoff and RMSE for basin_avg TWS (standarized scores)

Objective function of KGE for runoff and RMSE for basin_avg TWS (standarized scores)

Parameters

in	<i>real(dp), dimension(:) :: parameterset</i>	
in	<i>procedure(eval_interface) :: eval</i>	

Returns

real(dp) :: objective_kge_q_rmse_tws — objective function value (which will be e.g. minimized by an optimization routine like DDS)

Authors

Odrich Rakovec, Rohini Kumar

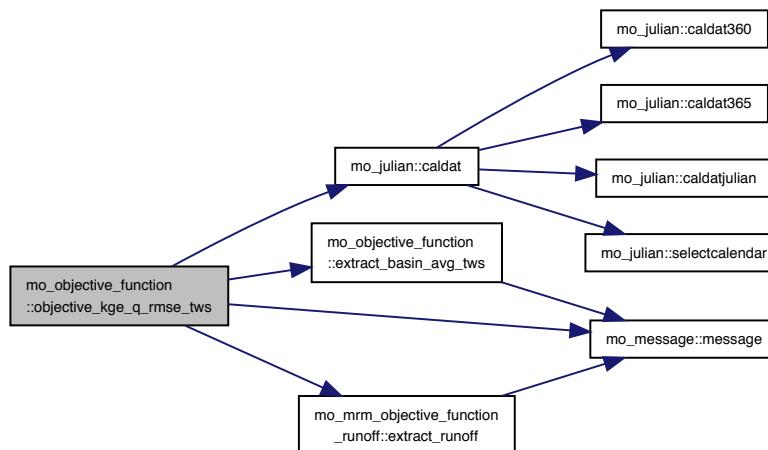
Date

Oct. 2015

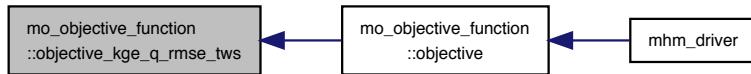
References *mo_julian::caldat()*, *mo_common_constants::eps_dp*, *mo_common_mhm_mrm_variables::evalper*, *extract_basin_avg_tws()*, *mo_mrm_objective_function_runoff::extract_runoff()*, *mo_message::message()*, *mo_common_variables::nbasins*, and *mo_common_constants::nodata_dp*.

Referenced by *objective()*.

Here is the call graph for this function:



Here is the caller graph for this function:



15.68.2.7 objective_kge_q_sm_corr()

```
real(dp) function mo_objective_function::objective_kge_q_sm_corr (
    real(dp), dimension(:), intent(in) parameterset,
    procedure(eval_interface), intent(in), pointer eval )
```

Objective function of KGE for runoff and correlation for SM.

Objective function of KGE for runoff and SSE for soil moisture (standardized scores). Further details can be found in the documentation of objective functions '14 - objective_multiple_gauges_kge_power6' and '13 - objective_sm_corr'.

Parameters

in	real(dp), dimension(:) :: parameterset	
in	procedure(eval_interface) :: eval	

Returns

real(dp) :: objective_kge_q_sse_sm — objective function value (which will be e.g. minimized by an optimization routine like DDS)

Authors

Matthias Zink

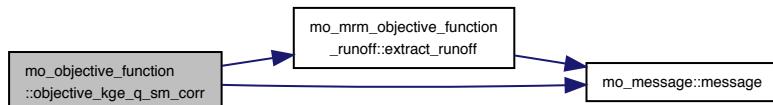
Date

Mar. 2017

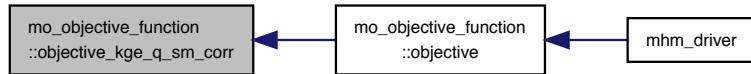
References mo_mrm_objective_function_runoff::extract_runoff(), mo_global_variables::l1_sm, mo_global_variables::l1_sm_mask, mo_common_variables::level1, mo_message::message(), and mo_common_variables::nbasins.

Referenced by objective().

Here is the call graph for this function:



Here is the caller graph for this function:



15.68.2.8 `objective_neutrons_kge_catchment_avg()`

```
real(dp) function mo_objective_function::objective_neutrons_kge_catchment_avg (
    real(dp), dimension(:), intent(in) parameterset,
    procedure(eval_interface), intent(in), pointer eval )
```

Objective function for neutrons.

The objective function only depends on a parameter vector. The model will be called with that parameter vector and the model output is subsequently compared to observed data. Therefore, the Kling-Gupta model efficiency KGE of the catchment average neutrons (N) is calculated

$$KGE = 1.0 - \sqrt{((1 - r)^2 + (1 - \alpha)^2 + (1 - \beta)^2)}$$

where r = Pearson product-moment CORRELATION coefficient α = ratio of simulated mean to observed mean SM β = ratio of simulated 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	<code>real(dp), dimension(:) :: parameterset</code>	
in	<code>procedure(eval_interface) :: eval</code>	

Returns

`real(dp) :: objective_neutrons_kge_catchment_avg` — objective function value (which will be e.g. minimized by an optimization routine)

Authors

Martin Schroen

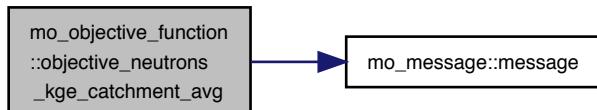
Date

Jun 2015

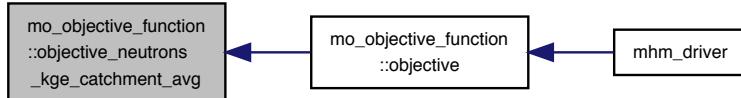
References `mo_global_variables::l1_neutronsdata`, `mo_global_variables::l1_neutronsdata_mask`, `mo_common_variables::level1`, `mo_message::message()`, `mo_common_variables::nbasins`, and `mo_common_constants::nodata_dp`.

Referenced by `objective()`.

Here is the call graph for this function:



Here is the caller graph for this function:



15.68.2.9 objective_sm_corr()

```

real(dp) function mo_objective_function::objective_sm_corr (
    real(dp), dimension(:), intent(in) parameterset,
    procedure(eval_interface), intent(in), pointer eval )

```

Objective function for soil moisture.

The objective function only depends on a parameter vector. The model will be called with that parameter vector and the model output is subsequently compared to observed data. Therefore the Pearson correlation between observed and modeled soil moisture on each grid cell j is compared

$$r_j = r^2(SM_{obs}^j, SM_{sim}^j)$$

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	<code>real(dp), dimension(:) :: parameterset</code>	
in	<code>procedure(eval_interface) :: eval</code>	

Returns

`real(dp) :: objective_sm_corr` — objective function value (which will be e.g. minimized by an optimization routine like DDS)

Authors

Matthias Zink

Date

March 2015

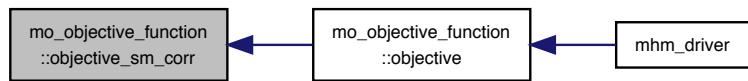
References `mo_global_variables::l1_sm`, `mo_global_variables::l1_sm_mask`, `mo_common_variables::level1`, `mo_message::message()`, and `mo_common_variables::nbasins`.

Referenced by `objective()`.

Here is the call graph for this function:



Here is the caller graph for this function:



15.68.2.10 objective_sm_kge_catchment_avg()

```
real(dp) function mo_objective_function::objective_sm_kge_catchment_avg (
    real(dp), dimension(:), intent(in) parameterset,
    procedure(eval_interface), intent(in), pointer eval )
```

Objective function for soil moisture.

The objective function only depends on a parameter vector. The model will be called with that parameter vector and the model output is subsequently compared to observed data. Therefore, the Kling-Gupta model efficiency KGE of the catchment average soil moisture (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 simulated standard deviation to observed standard deviation is calculated and the objective function for a given basin i is

$$\phi_i = 1.0 - KGE_i$$

ϕ_i is the objective since we always apply minimization methods. The minimal value of ϕ_i is 0 for the optimal KGE of 1.0. Finally, the overall objective function value OF is estimated based on the power-6 norm to combine the ϕ_i from all basins N .

$$OF = \sqrt[6]{\sum((1.0 - KGE_i)/N)^6}.$$

The observed data L1_sm, L1_sm_mask are global in this module.

Parameters

in	real(dp), dimension(:) :: parameterset	
in	procedure(eval_interface) :: eval	

Returns

real(dp) :: objective_sm_kge_catchment_avg — objective function value (which will be e.g. minimized by an optimization routine like DDS)

Authors

Matthias Zink

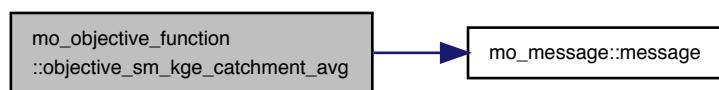
Date

May 2015

References mo_global_variables::l1_sm, mo_global_variables::l1_sm_mask, mo_common_variables::level1, mo_message::message(), mo_common_variables::nbasins, and mo_common_constants::nodata_dp.

Referenced by objective().

Here is the call graph for this function:



Here is the caller graph for this function:



15.68.2.11 objective_sm_pd()

```
real(dp) function mo_objective_function::objective_sm_pd (
    real(dp), dimension(:), intent(in) parameterset,
    procedure(eval_interface), intent(in), pointer eval )
```

Objective function for soil moisture.

The objective function only depends on a parameter vector. The model will be called with that parameter vector and the model output is subsequently compared to observed data. Therefore the Pattern Dissimilarity (PD) of observed and modeled soil moisture fields is calculated - aim: matching spatial patterns

$$E(t) = PD(SM_{obs}(t), SM_{sim}(t))$$

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 dissimilarity (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) :: objective_sm_pd — objective function value (which will be e.g. minimized by an optimization routine like DDS)

Authors

Matthias Zink

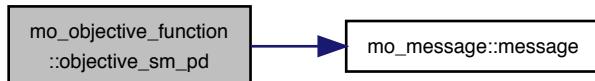
Date

May 2015

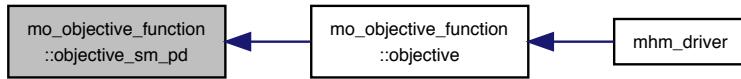
References mo_global_variables::l1_sm, mo_global_variables::l1_sm_mask, mo_common_variables::level1, mo_message::message(), mo_common_variables::nbasins, and mo_common_constants::nodata_dp.

Referenced by objective().

Here is the call graph for this function:



Here is the caller graph for this function:



15.68.2.12 objective_sm_sse_standard_score()

```
real(dp) function mo_objective_function::objective_sm_sse_standard_score (
    real(dp), dimension(:), intent(in) parameterset,
    procedure(eval_interface), intent(in), pointer eval )
```

Objective function for soil moisture.

The objective function only depends on a parameter vector. The model will be called with that parameter vector and the model output is subsequently compared to observed data. Therefore the sum of squared errors (SSE) of the standard score of observed and modeled soil moisture is calculated. The standard score or normalization (anomaly) make the objective function bias insensitive and basically the dynamics of the soil moisture is tried to capture by this objective 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 *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	<code>real(dp), dimension(:) :: parameterset</code>	
in	<code>procedure(eval_interface) :: eval</code>	

Returns

`real(dp) :: objective_sm_sse_standard_score` — objective function value (which will be e.g. minimized by an optimization routine like DDS)

Authors

Matthias Zink

Date

March 2015

References `mo_global_variables::l1_sm`, `mo_global_variables::l1_sm_mask`, `mo_common_variables::level1`, `mo_message::message()`, and `mo_common_variables::nbasins`.

Referenced by `objective()`.

Here is the call graph for this function:



Here is the caller graph for this function:



15.69 mo_optimization Module Reference

Wrapper subroutine for optimization against runoff and sm.

Functions/Subroutines

- subroutine, public **optimization** (eval, objective, dirConfigOut, funcBest, maskpara)
- Wrapper for optimization.*

15.69.1 Detailed Description

Wrapper subroutine for optimization against runoff and sm.

This module provides a wrapper subroutine for optimization of mRM/mHM against runoff or soil moisture.

Authors

Stephan Thober

Date

Oct 2015

15.69.2 Function/Subroutine Documentation

15.69.2.1 optimization()

```
subroutine, public mo_optimization::optimization (
    procedure(eval_interface), intent(in), pointer eval,
    procedure(objective_interface), intent(in), pointer objective,
    character(len = *), intent(in) dirConfigOut,
    real(dp), intent(out) funcBest,
    logical, dimension(:), intent(out), allocatable maskpara )
```

Wrapper for optimization.

This subroutine selects the optimization defined in a namelist, i.e. the global variable *opti_method*. It return the objective function value for a specific parameter set.

Parameters

in	<i>procedure(eval_interface) :: eval</i>	
in	<i>procedure(objective_interface) :: objective</i>	- objective function used in the optimization
in	<i>character(len = *) :: dirConfigOut</i>	- directory where to write ascii output
out	<i>real(dp) :: funcbest</i>	- best objective function value obtained during optimization
out	<i>logical, dimension(:) :: maskpara</i>	true = parameter will be optimized = parameter(i,4) = 1 false = parameter will not be optimized = parameter(i,4) = 0

Authors

Matthias Cuntz, Luis Samaniego, Juliane Mai, Matthias Zink and Stephan Thober

Date

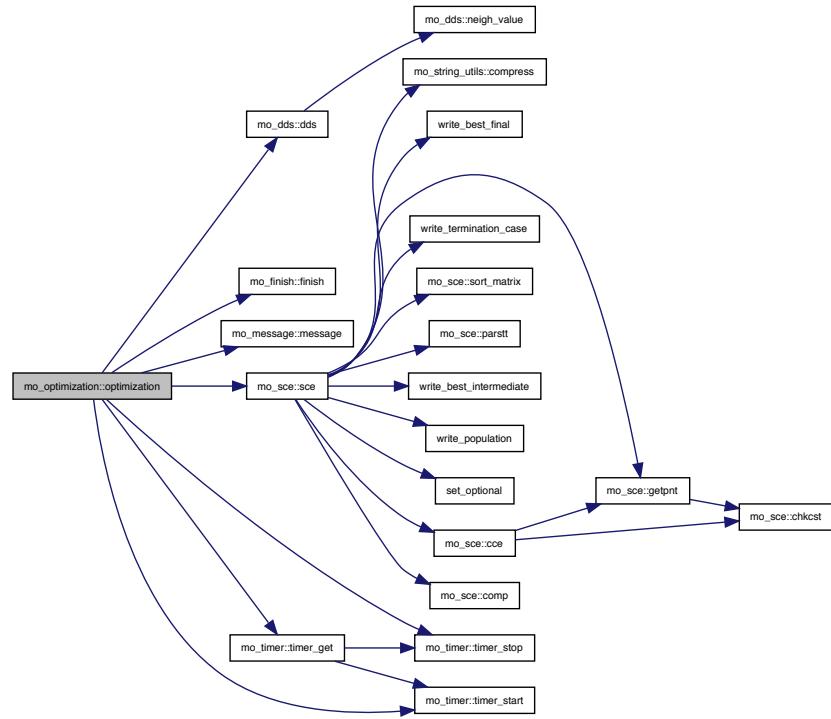
Oct 2015

References mo_dds::dds(), mo_common_mhm_mrm_variables::dds_r, mo_finish::finish(), mo_common_variables::global_parameters, mo_kind::i4, mo_kind::i8, mo_common_mhm_mrm_variables::mcmc_error_params, mo_common_mhm_mrm_variables::mcmc_opti, mo_message::message(), mo_common_mhm_mrm_variables::niterations, mo_common_mhm_mrm_variables::opti_function, mo_common_mhm_mrm_variables::opti_method, mo_common_mhm_mrm_variables::optimize_restart, mo_common_mhm_mrm_variables::sa_temp, mo_sce::sce(), mo_common_mhm_mrm_variables::sce_ngs, mo_common_mhm_mrm_variables::sce_npg,

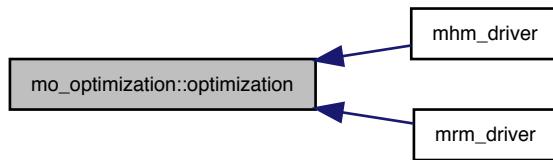
`mo_common_mhm_mrm_variables::sce_nps`, `mo_common_mhm_mrm_variables::seed`, `mo_timer::timer_get()`, `mo_timer::timer_start()`, and `mo_timer::timer_stop()`.

Referenced by `mhm_driver()`, and `mrm_driver()`.

Here is the call graph for this function:



Here is the caller graph for this function:



15.70 mo_optimization_utils Module Reference

Data Types

- interface `eval_interface`
- interface `objective_interface`

15.71 mo_orderpack Module Reference

Sort and ranking routines.

Data Types

- interface `ctrper`
- interface `fndnth`
- interface `indmed`
- interface `indnth`
- interface `inspar`
- interface `inssor`
- interface `mrgref`
- interface `mrgrnk`
- interface `mulcnt`
- interface `nearless`
- interface `omedian`
- interface `rapknr`
- interface `refpar`
- interface `refsor`
- interface `rinpar`
- interface `rnkpar`
- interface `sort`

Unconditional ranking.

- interface `sort_index`
- interface `uniinv`
- interface `unipar`
- interface `unirnk`
- interface `unista`
- interface `valmed`
- interface `valnth`

Functions/Subroutines

- integer(i4) function, dimension(size(arr)) `sort_index_dp` (arr)
- integer(i4) function, dimension(size(arr)) `sort_index_sp` (arr)
- integer(i4) function, dimension(size(arr)) `sort_index_i4` (arr)
- subroutine, private `d_ctrper` (XDONT, PCLS)
- subroutine, private `r_ctrper` (XDONT, PCLS)
- subroutine, private `i_ctrper` (XDONT, PCLS)
- real(kind=dp) function, private `d_fndnth` (XDONT, NORD)
- real(kind=sp) function, private `r_fndnth` (XDONT, NORD)
- integer(kind=i4) function, private `i_fndnth` (XDONT, NORD)
- subroutine, private `d_indmed` (XDONT, INDM)
- recursive subroutine, private `d_med` (XDATT, IDATT, ires_med)
- subroutine, private `r_indmed` (XDONT, INDM)
- recursive subroutine, private `r_med` (XDATT, IDATT, ires_med)
- subroutine, private `i_indmed` (XDONT, INDM)
- recursive subroutine, private `i_med` (XDATT, IDATT, ires_med)
- integer(kind=i4) function, private `d_indnth` (XDONT, NORD)
- integer(kind=i4) function, private `r_indnth` (XDONT, NORD)
- integer(kind=i4) function, private `i_indnth` (XDONT, NORD)
- subroutine, private `d_inspars` (XDONT, NORD)

- subroutine, private `r_inspar` (XDONT, NORD)
- subroutine, private `i_inspar` (XDONT, NORD)
- subroutine, private `d_inssor` (XDONT)
- subroutine, private `r_inssor` (XDONT)
- subroutine, private `i_inssor` (XDONT)
- real(kind=dp) function, private `d_median` (XDONT)
- real(kind=sp) function, private `r_median` (XDONT)
- integer(kind=i4) function, private `i_median` (XDONT)
- subroutine, private `d_mrgref` (XVALT, IRNGT)
- subroutine, private `r_mrgref` (XVALT, IRNGT)
- subroutine, private `i_mrgref` (XVALT, IRNGT)
- subroutine, private `d_mrgrnk` (XDONT, IRNGT)
- subroutine, private `r_mrgrnk` (XDONT, IRNGT)
- subroutine, private `i_mrgrnk` (XDONT, IRNGT)
- subroutine, private `d_mulcnt` (XDONT, IMULT)
- subroutine, private `r_mulcnt` (XDONT, IMULT)
- subroutine, private `i_mulcnt` (XDONT, IMULT)
- subroutine, private `d_rapknr` (XDONT, IRNGT, NORD)
- subroutine, private `r_rapknr` (XDONT, IRNGT, NORD)
- subroutine, private `i_rapknr` (XDONT, IRNGT, NORD)
- subroutine, private `d_refpar` (XDONT, IRNGT, NORD)
- subroutine, private `r_refpar` (XDONT, IRNGT, NORD)
- subroutine, private `i_refpar` (XDONT, IRNGT, NORD)
- subroutine, private `d_refsor` (XDONT)
- recursive subroutine, private `d_subSOR` (XDONT, IDEB1, IFIN1)
- subroutine, private `r_refsor` (XDONT)
- recursive subroutine, private `r_subSOR` (XDONT, IDEB1, IFIN1)
- subroutine, private `i_refsor` (XDONT)
- recursive subroutine, private `i_subSOR` (XDONT, IDEB1, IFIN1)
- subroutine, private `d_rinpar` (XDONT, IRNGT, NORD)
- subroutine, private `r_rinpar` (XDONT, IRNGT, NORD)
- subroutine, private `i_rinpar` (XDONT, IRNGT, NORD)
- subroutine, private `d_rnkpar` (XDONT, IRNGT, NORD)
- subroutine, private `r_rnkpar` (XDONT, IRNGT, NORD)
- subroutine, private `i_rnkpar` (XDONT, IRNGT, NORD)
- subroutine, private `d_uniinv` (XDONT, IGOEST)
- subroutine, private `r_uniinv` (XDONT, IGOEST)
- subroutine, private `i_uniinv` (XDONT, IGOEST)
- real(kind=dp) function, private `d_nearless` (XVAL)
- real(kind=sp) function, private `r_nearless` (XVAL)
- integer(kind=i4) function, private `i_nearless` (XVAL)
- subroutine, private `d_unipar` (XDONT, IRNGT, NORD)
- subroutine, private `r_unipar` (XDONT, IRNGT, NORD)
- subroutine, private `i_unipar` (XDONT, IRNGT, NORD)
- subroutine, private `d_unirnk` (XVALT, IRNGT, NUNI)
- subroutine, private `r_unirnk` (XVALT, IRNGT, NUNI)
- subroutine, private `i_unirnk` (XVALT, IRNGT, NUNI)
- subroutine, private `d_unista` (XDONT, NUNI)
- subroutine, private `r_unista` (XDONT, NUNI)
- subroutine, private `i_unista` (XDONT, NUNI)
- recursive real(kind=dp) function, private `d_valmed` (XDONT)
- recursive real(kind=sp) function, private `r_valmed` (XDONT)
- recursive integer(kind=i4) function, private `i_valmed` (XDONT)
- real(kind=dp) function, private `d_valnth` (XDONT, NORD)
- real(kind=sp) function, private `r_valnth` (XDONT, NORD)
- integer(kind=i4) function, private `i_valnth` (XDONT, NORD)

Variables

- `integer(kind=i4), dimension(:), allocatable, save idont`

15.71.1 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

15.71.2 Function/Subroutine Documentation

15.71.2.1 d_ctrper()

```
subroutine, private mo_orderpack::d_ctrper (
    real(kind = dp), dimension (:), intent(inout) XDONT,
    real(kind = dp), intent(in) PCLS ) [private]
```

15.71.2.2 d_fndnth()

```
real(kind = dp) function, private mo_orderpack::d_fndnth (
    real(kind = dp), dimension (:), intent(in) XDONT,
    integer(kind = i4), intent(in) NORD ) [private]
```

15.71.2.3 d_indmed()

```
subroutine, private mo_orderpack::d_indmed (
    real(kind = dp), dimension (:), intent(in) XDONT,
    integer(kind = i4), intent(out) INDN ) [private]
```

References `d_med()`, and `idont`.

Here is the call graph for this function:



15.71.2.4 d_indnth()

```
integer(kind = i4) function, private mo_orderpack::d_indnth (
    real(kind = dp), dimension (:), intent(in) XDONT,
    integer(kind = i4), intent(in) NORD ) [private]
```

15.71.2.5 d_inspar()

```
subroutine, private mo_orderpack::d_inspar (
    real(kind = dp), dimension (:), intent(inout) XDONT,
    integer(kind = i4), intent(in) NORD ) [private]
```

15.71.2.6 d_inssor()

```
subroutine, private mo_orderpack::d_inssor (
    real(kind = dp), dimension (:), intent(inout) XDONT ) [private]
```

Referenced by d_refsor().

Here is the caller graph for this function:



15.71.2.7 d_med()

```
recursive subroutine, private mo_orderpack::d_med (
    real(kind = dp), dimension (:), intent(in) XDATT,
    integer(kind = i4), dimension (:), intent(in) IDATT,
    integer(kind = i4), intent(out) ires_med ) [private]
```

Referenced by d_indmed().

Here is the caller graph for this function:



15.71.2.8 d_median()

```
real(kind = dp) function, private mo_orderpack::d_median (
    real(kind = dp), dimension (:), intent(in) XDONT ) [private]
```

15.71.2.9 d_mrgref()

```
subroutine, private mo_orderpack::d_mrgref (
    real(kind = dp), dimension (:), intent(in) XVALT,
    integer(kind = i4), dimension (:), intent(out) IRNGT ) [private]
```

15.71.2.10 d_mrgrnk()

```
subroutine, private mo_orderpack::d_mrgrnk (
    real(kind = dp), dimension (:), intent(in) XDONT,
    integer(kind = i4), dimension (:), intent(out) IRNGT ) [private]
```

15.71.2.11 d_mulcnt()

```
subroutine, private mo_orderpack::d_mulcnt (
    real(kind = dp), dimension (:), intent(in) XDONT,
    integer(kind = i4), dimension (:), intent(out) IMULT ) [private]
```

15.71.2.12 d_nearless()

```
real(kind = dp) function, private mo_orderpack::d_nearless (
    real(kind = dp), intent(in) XVAL ) [private]
```

15.71.2.13 d_rapknr()

```
subroutine, private mo_orderpack::d_rapknr (
    real(kind = dp), dimension (:), intent(in) XDONT,
```

```
integer(kind = i4), dimension (:), intent(out) IRNGT,
integer(kind = i4), intent(in) NORD ) [private]
```

15.71.2.14 d_refpar()

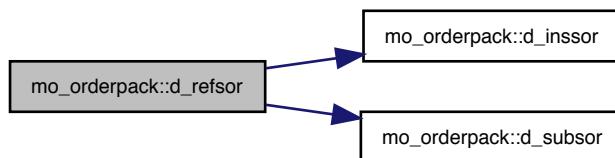
```
subroutine, private mo_orderpack::d_refpar (
    real(kind = dp), dimension (:), intent(in) XDONT,
    integer(kind = i4), dimension (:), intent(out) IRNGT,
    integer(kind = i4), intent(in) NORD ) [private]
```

15.71.2.15 d_refsor()

```
subroutine, private mo_orderpack::d_refsor (
    real(kind = dp), dimension (:), intent(inout) XDONT ) [private]
```

References d_inssor(), and d_substor().

Here is the call graph for this function:



15.71.2.16 d_rinpar()

```
subroutine, private mo_orderpack::d_rinpar (
    real(kind = dp), dimension (:), intent(in) XDONT,
    integer(kind = i4), dimension (:), intent(out) IRNGT,
    integer(kind = i4), intent(in) NORD ) [private]
```

15.71.2.17 d_rnkpar()

```
subroutine, private mo_orderpack::d_rnkpar (
    real(kind = dp), dimension (:), intent(in) XDONT,
    integer(kind = i4), dimension (:), intent(out) IRNGT,
    integer(kind = i4), intent(in) NORD ) [private]
```

15.71.2.18 d_subsol()

```
recursive subroutine, private mo_orderpack::d_subsol (
    real(kind = dp), dimension (:), intent(inout) XDONT,
    integer(kind = i4), intent(in) IDEBL,
    integer(kind = i4), intent(in) IFINI ) [private]
```

Referenced by d_refsor().

Here is the caller graph for this function:



15.71.2.19 d_uniinv()

```
subroutine, private mo_orderpack::d_uniinv (
    real(kind = dp), dimension (:), intent(in) XDONT,
    integer(kind = i4), dimension (:), intent(out) IGOEST ) [private]
```

15.71.2.20 d_unipar()

```
subroutine, private mo_orderpack::d_unipar (
    real(kind = dp), dimension (:), intent(in) XDONT,
    integer(kind = i4), dimension (:), intent(out) IRNGT,
    integer(kind = i4), intent(inout) NORD ) [private]
```

15.71.2.21 d_unirnk()

```
subroutine, private mo_orderpack::d_unirnk (
    real(kind = dp), dimension (:), intent(in) XVALT,
    integer(kind = i4), dimension (:), intent(out) IRNGT,
    integer(kind = i4), intent(out) NUNI ) [private]
```

15.71.2.22 d_unista()

```
subroutine, private mo_orderpack::d_unista (
    real(kind = dp), dimension (:), intent(inout) XDONT,
    integer(kind = i4), intent(out) NUNI ) [private]
```

15.71.2.23 d_valmed()

```
recursive real(kind = dp) function, private mo_orderpack::d_valmed (
    real(kind = dp), dimension (:), intent(in) XDONT )  [private]
```

15.71.2.24 d_valnth()

```
real(kind = dp) function, private mo_orderpack::d_valnth (
    real(kind = dp), dimension (:), intent(in) XDONT,
    integer(kind = i4), intent(in) NORD )  [private]
```

15.71.2.25 i_ctrper()

```
subroutine, private mo_orderpack::i_ctrper (
    integer(kind = i4), dimension (:), intent(inout) XDONT,
    real(kind = sp), intent(in) PCLS )  [private]
```

15.71.2.26 i_fndnth()

```
integer(kind = i4) function, private mo_orderpack::i_fndnth (
    integer(kind = i4), dimension (:), intent(in) XDONT,
    integer(kind = i4), intent(in) NORD )  [private]
```

15.71.2.27 i_ndmed()

```
subroutine, private mo_orderpack::i_ndmed (
    integer(kind = i4), dimension (:), intent(in) XDONT,
    integer(kind = i4), intent(out) INDN )  [private]
```

References i_med(), and idont.

Here is the call graph for this function:



15.71.2.28 i_indnth()

```
integer(kind = i4) function, private mo_orderpack::i_indnth (
    integer(kind = i4), dimension (:), intent(in) XDONT,
```

```
integer(kind = i4), intent(in) NORD ) [private]
```

15.71.2.29 i_inspar()

```
subroutine, private mo_orderpack::i_inspar (
    integer(kind = i4), dimension (:), intent(inout) XDONT,
    integer(kind = i4), intent(in) NORD ) [private]
```

15.71.2.30 i_inssor()

```
subroutine, private mo_orderpack::i_inssor (
    integer(kind = i4), dimension (:), intent(inout) XDONT ) [private]
```

Referenced by i_refsor().

Here is the caller graph for this function:



15.71.2.31 i_med()

```
recursive subroutine, private mo_orderpack::i_med (
    integer(kind = i4), dimension (:), intent(in) XDATT,
    integer(kind = i4), dimension (:), intent(in) IDATT,
    integer(kind = i4), intent(out) ires_med ) [private]
```

Referenced by i_indmed().

Here is the caller graph for this function:



15.71.2.32 i_median()

```
integer(kind = i4) function, private mo_orderpack::i_median (
    integer(kind = i4), dimension (:), intent(in) XDONT ) [private]
```

15.71.2.33 i_mrgref()

```
subroutine, private mo_orderpack::i_mrgref (
    integer(kind = i4), dimension (:), intent(in) XVALT,
    integer(kind = i4), dimension (:), intent(out) IRNGT ) [private]
```

15.71.2.34 i_mrgrnk()

```
subroutine, private mo_orderpack::i_mrgrnk (
    integer(kind = i4), dimension (:), intent(in) XDONT,
    integer(kind = i4), dimension (:), intent(out) IRNGT ) [private]
```

15.71.2.35 i_mulcnt()

```
subroutine, private mo_orderpack::i_mulcnt (
    integer(kind = i4), dimension (:), intent(in) XDONT,
    integer(kind = i4), dimension (:), intent(out) IMULT ) [private]
```

15.71.2.36 i_nearless()

```
integer(kind = i4) function, private mo_orderpack::i_nearless (
    integer(kind = i4), intent(in) XVAL ) [private]
```

15.71.2.37 i_rapknr()

```
subroutine, private mo_orderpack::i_rapknr (
    integer(kind = i4), dimension (:), intent(in) XDONT,
    integer(kind = i4), dimension (:), intent(out) IRNGT,
    integer(kind = i4), intent(in) NORD ) [private]
```

15.71.2.38 i_refpar()

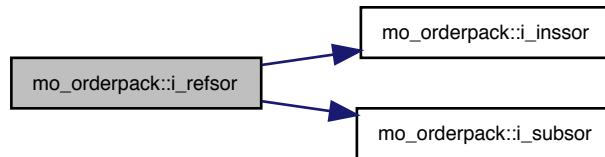
```
subroutine, private mo_orderpack::i_refpar (
    integer(kind = i4), dimension (:), intent(in) XDONT,
    integer(kind = i4), dimension (:), intent(out) IRNGT,
    integer(kind = i4), intent(in) NORD ) [private]
```

15.71.2.39 i_refsor()

```
subroutine, private mo_orderpack::i_refsor (
    integer(kind = i4), dimension (:), intent(inout) XDONT ) [private]
```

References i_inssor(), and i_substor().

Here is the call graph for this function:



15.71.2.40 i_rinpar()

```
subroutine, private mo_orderpack::i_rinpar (
    integer(kind = i4), dimension (:), intent(in) XDONT,
    integer(kind = i4), dimension (:), intent(out) IRNGT,
    integer(kind = i4), intent(in) NORD ) [private]
```

15.71.2.41 i_rnkpar()

```
subroutine, private mo_orderpack::i_rnkpar (
    integer(kind = i4), dimension (:), intent(in) XDONT,
    integer(kind = i4), dimension (:), intent(out) IRNGT,
    integer(kind = i4), intent(in) NORD ) [private]
```

15.71.2.42 i_substor()

```
recursive subroutine, private mo_orderpack::i_substor (
    integer(kind = i4), dimension (:), intent(inout) XDONT,
    integer(kind = i4), intent(in) IDEB1,
    integer(kind = i4), intent(in) IFINI ) [private]
```

Referenced by i_refsor().

Here is the caller graph for this function:



15.71.2.43 i_uniinv()

```
subroutine, private mo_orderpack::i_uniinv (
    integer(kind = i4), dimension (:), intent(in) XDONT,
    integer(kind = i4), dimension (:), intent(out) IGOEST ) [private]
```

15.71.2.44 i_unipar()

```
subroutine, private mo_orderpack::i_unipar (
    integer(kind = i4), dimension (:), intent(in) XDONT,
    integer(kind = i4), dimension (:), intent(out) IRNGT,
    integer(kind = i4), intent(inout) NORD ) [private]
```

15.71.2.45 i_unirnk()

```
subroutine, private mo_orderpack::i_unirnk (
    integer(kind = i4), dimension (:), intent(in) XVALT,
    integer(kind = i4), dimension (:), intent(out) IRNGT,
    integer(kind = i4), intent(out) NUNI ) [private]
```

15.71.2.46 i_unista()

```
subroutine, private mo_orderpack::i_unista (
    integer(kind = i4), dimension (:), intent(inout) XDONT,
    integer(kind = i4), intent(out) NUNI ) [private]
```

15.71.2.47 i_valmed()

```
recursive integer(kind = i4) function, private mo_orderpack::i_valmed (
    integer(kind = i4), dimension (:), intent(in) XDONT ) [private]
```

15.71.2.48 i_valnth()

```
integer(kind = i4) function, private mo_orderpack::i_valnth (
    integer(kind = i4), dimension (:), intent(in) XDONT,
    integer(kind = i4), intent(in) NORD ) [private]
```

15.71.2.49 r_ctrper()

```
subroutine, private mo_orderpack::r_ctrper (
    real(kind = sp), dimension (:), intent(inout) XDONT,
    real(kind = sp), intent(in) PCLS ) [private]
```

15.71.2.50 r_fndnth()

```
real(kind = sp) function, private mo_orderpack::r_fndnth (
    real(kind = sp), dimension (:), intent(in) XDONT,
    integer(kind = i4), intent(in) NORD ) [private]
```

15.71.2.51 r_indmed()

```
subroutine, private mo_orderpack::r_indmed (
    real(kind = sp), dimension (:), intent(in) XDONT,
    integer(kind = i4), intent(out) INDIM ) [private]
```

References idont, and r_med().

Here is the call graph for this function:



15.71.2.52 r_indnth()

```
integer(kind = i4) function, private mo_orderpack::r_indnth (
    real(kind = sp), dimension (:), intent(in) XDONT,
    integer(kind = i4), intent(in) NORD ) [private]
```

15.71.2.53 r_inspar()

```
subroutine, private mo_orderpack::r_inspar (
```

```
real(kind = sp), dimension (:), intent(inout) XDONT,
integer(kind = i4), intent(in) NORD ) [private]
```

15.71.2.54 r_inssor()

```
subroutine, private mo_orderpack::r_inssor (
    real(kind = sp), dimension (:), intent(inout) XDONT ) [private]
```

Referenced by r_refsor().

Here is the caller graph for this function:



15.71.2.55 r_med()

```
recursive subroutine, private mo_orderpack::r_med (
    real(kind = sp), dimension (:), intent(in) XDATT,
    integer(kind = i4), dimension (:), intent(in) IDATT,
    integer(kind = i4), intent(out) ires_med ) [private]
```

Referenced by r_indmed().

Here is the caller graph for this function:



15.71.2.56 r_median()

```
real(kind = sp) function, private mo_orderpack::r_median (
    real(kind = sp), dimension (:), intent(in) XDONT ) [private]
```

15.71.2.57 r_mrgref()

```
subroutine, private mo_orderpack::r_mrgref (
    real(kind = sp), dimension (:), intent(in) XVALT,
    integer(kind = i4), dimension (:), intent(out) IRNGT )  [private]
```

15.71.2.58 r_mrgrnk()

```
subroutine, private mo_orderpack::r_mrgrnk (
    real(kind = sp), dimension (:), intent(in) XDONT,
    integer(kind = i4), dimension (:), intent(out) IRNGT )  [private]
```

15.71.2.59 r_mulcnt()

```
subroutine, private mo_orderpack::r_mulcnt (
    real(kind = sp), dimension (:), intent(in) XDONT,
    integer(kind = i4), dimension (:), intent(out) IMULT )  [private]
```

15.71.2.60 r_nearless()

```
real(kind = sp) function, private mo_orderpack::r_nearless (
    real(kind = sp), intent(in) XVAL )  [private]
```

15.71.2.61 r_rapknr()

```
subroutine, private mo_orderpack::r_rapknr (
    real(kind = sp), dimension (:), intent(in) XDONT,
    integer(kind = i4), dimension (:), intent(out) IRNGT,
    integer(kind = i4), intent(in) NORD )  [private]
```

15.71.2.62 r_refpar()

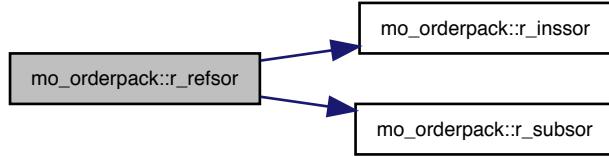
```
subroutine, private mo_orderpack::r_refpar (
    real(kind = sp), dimension (:), intent(in) XDONT,
    integer(kind = i4), dimension (:), intent(out) IRNGT,
    integer(kind = i4), intent(in) NORD )  [private]
```

15.71.2.63 r_refsor()

```
subroutine, private mo_orderpack::r_refsor (
    real(kind = sp), dimension (:), intent(inout) XDONT )  [private]
```

References r_inssor(), and r_subssor().

Here is the call graph for this function:



15.71.2.64 r_rinpar()

```

subroutine, private mo_orderpack::r_rinpar (
    real(kind = sp), dimension (:), intent(in) XDONT,
    integer(kind = i4), dimension (:), intent(out) IRNGT,
    integer(kind = i4), intent(in) NORD ) [private]
  
```

15.71.2.65 r_rnkpar()

```

subroutine, private mo_orderpack::r_rnkpar (
    real(kind = sp), dimension (:), intent(in) XDONT,
    integer(kind = i4), dimension (:), intent(out) IRNGT,
    integer(kind = i4), intent(in) NORD ) [private]
  
```

15.71.2.66 r_substor()

```

recursive subroutine, private mo_orderpack::r_substor (
    real(kind = sp), dimension (:), intent(inout) XDONT,
    integer(kind = i4), intent(in) IDEB1,
    integer(kind = i4), intent(in) IFINI ) [private]
  
```

Referenced by r_refsor().

Here is the caller graph for this function:



15.71.2.67 r_uniinv()

```
subroutine, private mo_orderpack::r_uniinv (
    real(kind = sp), dimension (:), intent(in) XDONT,
    integer(kind = i4), dimension (:), intent(out) IGOEST )  [private]
```

15.71.2.68 r_unipar()

```
subroutine, private mo_orderpack::r_unipar (
    real(kind = sp), dimension (:), intent(in) XDONT,
    integer(kind = i4), dimension (:), intent(out) IRNGT,
    integer(kind = i4), intent(inout) NORD )  [private]
```

15.71.2.69 r_unirnk()

```
subroutine, private mo_orderpack::r_unirnk (
    real(kind = sp), dimension (:), intent(in) XVALT,
    integer(kind = i4), dimension (:), intent(out) IRNGT,
    integer(kind = i4), intent(out) NUNI )  [private]
```

15.71.2.70 r_unista()

```
subroutine, private mo_orderpack::r_unista (
    real(kind = sp), dimension (:), intent(inout) XDONT,
    integer(kind = i4), intent(out) NUNI )  [private]
```

15.71.2.71 r_valmed()

```
recursive real(kind = sp) function, private mo_orderpack::r_valmed (
    real(kind = sp), dimension (:), intent(in) XDONT )  [private]
```

15.71.2.72 r_valnth()

```
real(kind = sp) function, private mo_orderpack::r_valnth (
    real(kind = sp), dimension (:), intent(in) XDONT,
    integer(kind = i4), intent(in) NORD )  [private]
```

15.71.2.73 sort_index_dp()

```
integer(i4) function, dimension(size(arr)) mo_orderpack::sort_index_dp (
    real(dp), dimension(:), intent(in) arr )  [private]
```

15.71.2.74 sort_index_i4()

```
integer(i4) function, dimension(size(arr)) mo_orderpack::sort_index_i4 (
    integer(i4), dimension(:), intent(in) arr ) [private]
```

15.71.2.75 sort_index_sp()

```
integer(i4) function, dimension(size(arr)) mo_orderpack::sort_index_sp (
    real(sp), dimension(:), intent(in) arr ) [private]
```

15.71.3 Variable Documentation

15.71.3.1 idont

```
integer(kind = i4), dimension(:), allocatable, save mo_orderpack::idont [private]
```

Referenced by `d_indmed()`, `i_indmed()`, and `r_indmed()`.

15.72 mo_percentile Module Reference

Data Types

- interface `median`
- interface `n_element`
- interface `percentile`
- interface `qmedian`

Functions/Subroutines

- real(dp) function `median_dp` (arrin, mask)
- real(sp) function `median_sp` (arrin, mask)
- real(dp) function `n_element_dp` (idat, n, mask, before, after, previous, next)
- real(sp) function `n_element_sp` (idat, n, mask, before, after, previous, next)
- real(dp) function `percentile_0d_dp` (arrin, k, mask, mode_in)
- real(sp) function `percentile_0d_sp` (arrin, k, mask, mode_in)
- real(dp) function, dimension(size(k)) `percentile_1d_dp` (arrin, k, mask, mode_in)
- real(sp) function, dimension(size(k)) `percentile_1d_sp` (arrin, k, mask, mode_in)
- real(dp) function `qmedian_dp` (dat)
- real(sp) function `qmedian_sp` (dat)

15.72.1 Function/Subroutine Documentation

15.72.1.1 median_dp()

```
real(dp) function mo_percentile::median_dp (
    real(dp), dimension(:), intent(in) arrin,
    logical, dimension(:), intent(in), optional mask ) [private]
```

15.72.1.2 median_sp()

```
real(sp) function mo_percentile::median_sp (
    real(sp), dimension(:), intent(in) arrin,
    logical, dimension(:), intent(in), optional mask ) [private]
```

15.72.1.3 n_element_dp()

```
real(dp) function mo_percentile::n_element_dp (
    real(dp), dimension(:), intent(in) idat,
    integer(i4), intent(in) n,
    logical, dimension(:), intent(in), optional mask,
    real(dp), intent(out), optional before,
    real(dp), intent(out), optional after,
    real(dp), intent(out), optional previous,
    real(dp), intent(out), optional next ) [private]
```

15.72.1.4 n_element_sp()

```
real(sp) function mo_percentile::n_element_sp (
    real(sp), dimension(:), intent(in) idat,
    integer(i4), intent(in) n,
    logical, dimension(:), intent(in), optional mask,
    real(sp), intent(out), optional before,
    real(sp), intent(out), optional after,
    real(sp), intent(out), optional previous,
    real(sp), intent(out), optional next ) [private]
```

15.72.1.5 percentile_0d_dp()

```
real(dp) function mo_percentile::percentile_0d_dp (
    real(dp), dimension(:), intent(in) arrin,
    real(dp), intent(in) k,
    logical, dimension(:), intent(in), optional mask,
    integer(i4), intent(in), optional mode_in ) [private]
```

References mo_kind::i4.

15.72.1.6 percentile_0d_sp()

```
real(sp) function mo_percentile::percentile_0d_sp (
    real(sp), dimension(:), intent(in) arrin,
```

```
real(sp), intent(in) k,
logical, dimension(:), intent(in), optional mask,
integer(i4), intent(in), optional mode_in ) [private]
```

References mo_kind::i4.

15.72.1.7 percentile_1d_dp()

```
real(dp) function, dimension(size(k)) mo_percentile::percentile_1d_dp (
    real(dp), dimension(:), intent(in) arrin,
    real(dp), dimension(:), intent(in) k,
    logical, dimension(:), intent(in), optional mask,
    integer(i4), intent(in), optional mode_in ) [private]
```

References mo_kind::i4.

15.72.1.8 percentile_1d_sp()

```
real(sp) function, dimension(size(k)) mo_percentile::percentile_1d_sp (
    real(sp), dimension(:), intent(in) arrin,
    real(sp), dimension(:), intent(in) k,
    logical, dimension(:), intent(in), optional mask,
    integer(i4), intent(in), optional mode_in ) [private]
```

References mo_kind::i4.

15.72.1.9 qmedian_dp()

```
real(dp) function mo_percentile::qmedian_dp (
    real(dp), dimension(:), intent(inout) dat ) [private]
```

15.72.1.10 qmedian_sp()

```
real(sp) function mo_percentile::qmedian_sp (
    real(sp), dimension(:), intent(inout) dat ) [private]
```

15.73 mo_pet Module Reference

Module for calculating reference/potential evapotranspiration [mm s-1].

Functions/Subroutines

- elemental pure real(dp) function, public [pet_hargreaves](#) (HarSamCoeff, HarSamConst, tavg, tmax, tmin, latitude, doy)

Reference Evapotranspiration after Hargreaves.

- elemental pure real(dp) function, public [pet_priestly](#) (PrieTayParam, Rn, tavg)

Reference Evapotranspiration after Priestly-Taylor.

- elemental pure real(dp) function, public `pet_penman` (net_rad, tavg, act_vap_pressure, aerodyn_resistance, bulksurface_resistance, a_s, a_sh)
Reference Evapotranspiration after Penman-Monteith.
- elemental pure real(dp) function, private `extraterr_rad_approx` (doy, latitude)
Approximation of extraterrestrial radiation.
- elemental pure real(dp) function, private `slope_satpressure` (tavg)
slope of saturation vapour pressure curve
- elemental pure real(dp) function, private `sat_vap_pressure` (tavg)
calculation of the saturation vapour pressure

15.73.1 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

15.73.2 Function/Subroutine Documentation

15.73.2.1 extraterr_rad_approx()

```
elemental pure real(dp) function, private mo_pet::extraterr_rad_approx (
    integer(i4), intent(in) doy,
    real(dp), intent(in) latitude ) [private]
```

Approximation of extraterrestrial radiation.

Approximation of extraterrestrial radiation at the top of the atmosphere R_a after Duffie and Beckman (1980). R_a is converted from $[J m^{-2} d^{-1}]$ in $[mm d^{-1}]$.

$$R_a = \frac{86400}{\pi \cdot \lambda} \cdot E_0 \cdot d_r \cdot (\omega \cdot \sin(\text{latitude}) \cdot \sin(\delta) + \cos(\text{latitude}) \cdot \cos(\delta) \cdot \sin(\omega))$$

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:

$$d_r = 1 + 0.033 \cdot \cos\left(\frac{2 \cdot \pi \cdot \text{doy}}{365}\right)$$

in which doy is the day of the year. The solar declination [radians] defined by

$$\delta = 0.4093 \cdot \sin\left(\frac{2 \cdot \pi \cdot \text{doy}}{365} - 1.405\right)$$

The sunset hour angle [radians]:

$$\omega = \arccos(-\tan(\text{latitude}) * \tan(\delta))$$

$< \lambda = 2.45 \cdot 10^6 J m^{-2} mm^{-1}$ is the latent heat of vaporization.

Parameters

in	<i>integer(i4) :: doy</i>	day of year [-]
in	<i>real(dp) :: latitude</i>	latitude [rad]

Returns

real(dp) :: extraterr_rad_approx — extraterrestrial radiation approximation [$W\ m^{-2}$]

Authors

Matthias Zink

Date

Apr 2014

References *mo_common_constants::daysecs*, *mo_mhm_constants::duffiedelta1*, *mo_mhm_constants::duffiedelta2*, *mo_mhm_constants::duffiedr*, *mo_constants::pi_d*, *mo_constants::solarconst_dp*, *mo_constants::specheatet_dp*, *mo_constants::twopi_d*, and *mo_common_constants::yeardays*.

Referenced by *pet_hargreaves()*.

Here is the caller graph for this function:

15.73.2.2 *pet_hargreaves()*

```

elemental pure real(dp) function, public mo_pet::pet_hargreaves (
    real(dp), intent(in) HarSamCoeff,
    real(dp), intent(in) HarSamConst,
    real(dp), intent(in) tavg,
    real(dp), intent(in) tmax,
    real(dp), intent(in) tmin,
    real(dp), intent(in) latitude,
    integer(i4), intent(in) doy )
  
```

Reference Evapotranspiration after Hargreaves.

Calculates the Reference Evapotranspiration [$mm\ d^{-1}$] based on the Hargreaves-Samani (1982) model for a given cell by applying the equation

$$PET = HarSamCoeff * R_a * (T_{avg} + HarSamConst) * \sqrt{T_{max} - T_{min}}$$

where R_a [$W\ m^{-2}$] is the incoming solar radiation and T_{avg} , T_{max} and T_{min} [$^{\circ}C$] are the mean, maximum, and minimum daily temperatures at the given day, respectively.

Note

Hargreaves, G.H., and Samani, Z.A. (1982). "Estimating potential evapotranspiration." Tech. Note, J. Irrig. and drain. Engrg., ASCE, 108(3):225-230.

Parameters

in	<i>real(dp) :: HarSamCoeff</i>	coefficient of Hargreaves-Samani equation [-]
in	<i>real(dp) :: HarSamConst</i>	constant of Hargreaves-Samani equation [-]
in	<i>real(dp) :: tavg</i>	daily mean temperature [$^{\circ}\text{C}$]
in	<i>real(dp) :: tmax</i>	daily maximum of temp [$^{\circ}\text{C}$]
in	<i>real(dp) :: tmin</i>	daily minimum of temp [$^{\circ}\text{C}$]
in	<i>real(dp) :: latitude</i>	latitude of the cell for Ra estimation [radians]
in	<i>integer(i4) :: doy</i>	day of year for Ra estimation

Returns

real(dp) :: pet_hargreaves — Hargreaves-Samani pot. evapotranspiration [mm s-1]

Authors

Matthias Zink

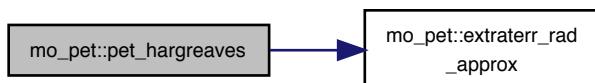
Date

Dec 2012

References *mo_constants::deg2rad_dp*, and *extraterr_rad_approx()*.

Referenced by *mo_mhm::mhm()*.

Here is the call graph for this function:



Here is the caller graph for this function:



15.73.2.3 pet_penman()

```
elemental pure real(dp) function, public mo_pet::pet_penman (
    real(dp), intent(in) net_rad,
```

```

real(dp), intent(in) tavg,
real(dp), intent(in) act_vap_pressure,
real(dp), intent(in) aerodyn_resistance,
real(dp), intent(in) bulksurface_resistance,
real(dp), intent(in) a_s,
real(dp), intent(in) a_sh )

```

Reference Evapotranspiration after Penman-Monteith.

Calculates the reference evapotranspiration [$mm\ d^{-1}$] based on the Penman-Monteith model for every given cell by applying the equation

$$PET = \frac{1}{\lambda} \cdot \frac{\Delta \cdot R_n + \rho \cdot c_p \cdot (e_s - e) \cdot \frac{a_{sh}}{r_a}}{\Delta + \gamma \cdot \frac{a_{sh}}{a_s} \cdot \left(1 + \frac{r_s}{r_a}\right)}$$

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, γ [$kPa\ K^{-1}$] 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-Monteith 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-Monteith 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 [$W\ m^{-2}$]
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

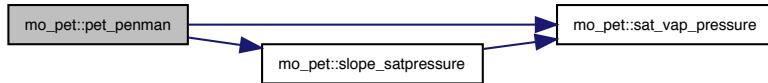
Date

Apr 2014

References mo_constants::cp0_dp, mo_common_constants::daysecs, mo_constants::psychro_dp, mo_constants::rho0_dp, sat_vap_pressure(), slope_satpressure(), and mo_constants::specheatet_dp.

Referenced by mo_mhm::mhm().

Here is the call graph for this function:



Here is the caller graph for this function:



15.73.2.4 pet_priestly()

```
elemental pure real(dp) function, public mo_pet:::pet_priestly (
    real(dp), intent(in) PrieTayParam,
    real(dp), intent(in) Rn,
    real(dp), intent(in) tavg )
```

Reference Evapotranspiration after Priestly-Taylor.

Calculates the Reference Evapotranspiration [$mm\ d^{-1}$] based on the Priestly-Taylor (1972) model for every given cell by applying the equation

$$PET = \alpha * \frac{\Delta}{(\gamma + \Delta)} * R_n$$

where R_n [$W\ m^{-2}$] is the net solar radiation $\Delta = f(T_{avg})$ is the slope of the saturation-vapour pressure curve and α is a emperical coefficient.

Parameters

in	<code>real(dp) :: PrieTayParam</code>	Priestley-Taylor coefficient $\alpha[-]$
in	<code>real(dp) :: Rn</code>	net solar radiation [$W\ m^{-2}$]
in	<code>real(dp) :: Tavg</code>	

Returns

`real(dp) :: pet_priestly` — Priestley-Taylor pot. evapotranspiration [$mm\ s^{-1}$]

Authors

Matthias Zink

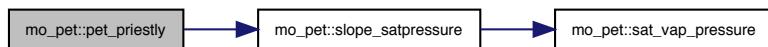
Date

Apr 2014

References `mo_common_constants::daysecs`, `mo_constants::psychro_dp`, `slope_satpressure()`, and `mo_constants::specheatet_dp`.

Referenced by `mo_mhm::mhm()`.

Here is the call graph for this function:



Here is the caller graph for this function:



15.73.2.5 `sat_vap_pressure()`

```
elemental pure real(dp) function, private mo_pet::sat_vap_pressure (
    real(dp), intent(in) tavg ) [private]
```

calculation of the saturation vapour pressure

Calculation of the saturation vapour pressure

$$e_s(T_a) = 0.6108 \cdot \exp\left(\frac{17.27 \cdot T_a}{T_a + 237.3}\right)$$

Parameters

in	<code>real(dp) :: tavg</code>	temperature [degC]
----	-------------------------------	--------------------

Returns

`real(dp) :: sat_vap_pressure` — saturation vapour pressure [kPa]

Authors

Matthias Zink

Date

Apr 2014

References mo_mhm_constants::tetens_c1, mo_mhm_constants::tetens_c2, and mo_mhm_constants::tetens_c3.

Referenced by pet_penman(), and slope_satpressure().

Here is the caller graph for this function:



15.73.2.6 slope_satpressure()

```
elemental pure real(dp) function, private mo_pet::slope_satpressure (
    real(dp), intent(in) tavg ) [private]
```

slope of saturation vapour pressure curve

slope of saturation vapour pressure curve after Tetens

$$\Delta = \frac{0.6108 * e_s(T_a)}{e^{(2 \cdot \log(T_a + 237.3))}}$$

Parameters

in	real(dp) :: tavg	average daily temperature [°C]
----	------------------	--------------------------------

Returns

real(dp) :: slope_satpressure — slope of saturation vapour pressure curve [kPa K-1]

Authors

Matthias Zink

Date

Apr 2014

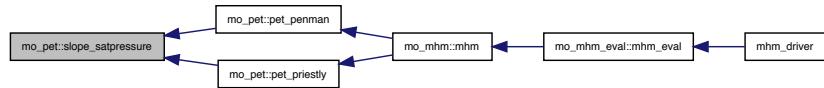
References `sat_vap_pressure()`, `mo_mhm_constants::satpressureslope1`, and `mo_mhm_constants::tetens_c3`.

Referenced by `pet_penman()`, and `pet_priestly()`.

Here is the call graph for this function:



Here is the caller graph for this function:



15.74 mo_prepare_gridded_lai Module Reference

Prepare daily LAI fields (e.g., MODIS data) for mHM.

Functions/Subroutines

- subroutine, public `prepare_gridded_daily_lai_data` (iBasin, nrows, ncols, mask, LAIPer_iBasin)
Prepare gridded daily LAI data.
- subroutine, public `prepare_gridded_mean_monthly_lai_data` (iBasin, nrows, ncols, mask)
prepare_gridded_mean_monthly_LAI_data

15.74.1 Detailed Description

Prepare daily LAI fields (e.g., MODIS data) for mHM.

Prepare daily LAI fields (e.g., MODIS data) for mHM

Authors

John Craven & Rohini Kumar

Date

Aug 2013

15.74.2 Function/Subroutine Documentation

15.74.2.1 prepare_gridded_daily_lai_data()

```
subroutine, public mo_prepare_gridded_lai::prepare_gridded_daily_lai_data (
    integer(i4), intent(in) iBasin,
    integer(i4), intent(in) nrows,
    integer(i4), intent(in) ncols,
    logical, dimension(:, :), intent(in) mask,
    type(period), intent(in), optional LAIPer_iBasin )
```

Prepare gridded daily LAI data.

Prepare gridded daily LAI data at Level-0 (e.g., using MODIS datasets)

Parameters

in	integer(i4) :: iBasin, nrows, ncols	Basin Id
in	integer(i4) :: iBasin, nrows, ncols	Basin Id
in	integer(i4) :: iBasin, nrows, ncols	Basin Id
in	logical, dimension(:, :) :: mask	
in	type(period), optional :: LAIPer_iBasin	

Authors

John Craven & Rohini Kumar

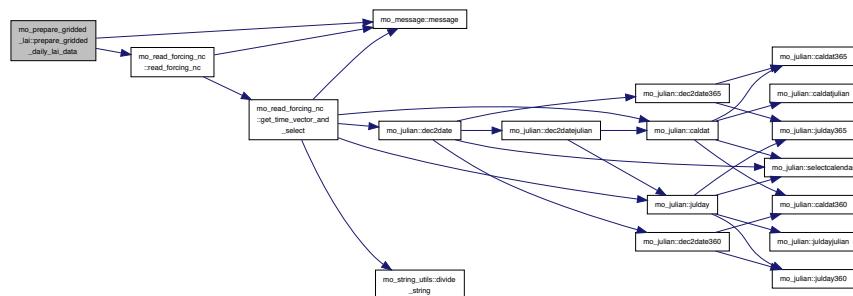
Date

Aug 2013

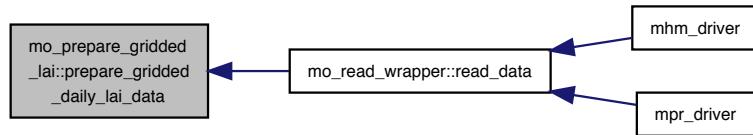
References mo_mpr_global_variables::dirgridded_lai, mo_mpr_global_variables::inputformat_gridded_lai, mo_mpr_global_variables::l0_gridded_lai, mo_message::message(), mo_mpr_global_variables::nlai, mo_read_forcing_nc::read_forcing_nc(), and mo_mpr_global_variables::timestep_lai_input.

Referenced by mo_read_wrapper::read_data().

Here is the call graph for this function:



Here is the caller graph for this function:



15.74.2.2 prepare_gridded_mean_monthly_lai_data()

```

subroutine, public mo_prepare_gridded_lai::prepare_gridded_mean_monthly_lai_data (
    integer(i4), intent(in) iBasin,
    integer(i4), intent(in) nrows,
    integer(i4), intent(in) ncols,
    logical, dimension(:, :, ), intent(in) mask )
  
```

`prepare_gridded_mean_monthly_LAI_data`

Long term mean monthly gridded LAI data at Level-0 (e.g., using MODIS datasets) The netcdf file should contain 12 (calender months) gridded fields of climatological LAI data at the input L0 data resolution.

Parameters

in	<code>integer(i4) :: iBasin, nrows, ncols</code>	Basin Id
in	<code>integer(i4) :: iBasin, nrows, ncols</code>	Basin Id
in	<code>integer(i4) :: iBasin, nrows, ncols</code>	Basin Id
in	<code>logical, dimension(:, :,) :: mask</code>	

Authors

Rohini Kumar

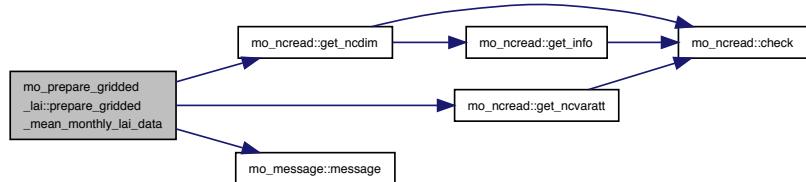
Date

Dec 2016

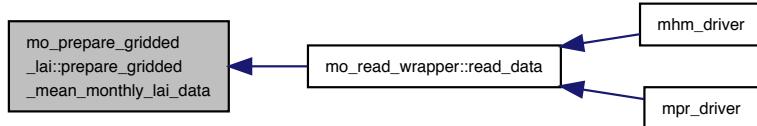
References mo_mpr_global_variables::dirgridded_lai, mo_ncread::get_ncdim(), mo_ncread::get_ncvaratt(), mo_mpr_global_variables::l0_gridded_lai, mo_message::message(), and mo_mpr_global_variables::nlai.

Referenced by mo_read_wrapper::read_data().

Here is the call graph for this function:



Here is the caller graph for this function:



15.75 mo_read_forcing_nc Module Reference

Reads forcing input data.

Functions/Subroutines

- subroutine, public [read_forcing_nc](#) (folder, nRows, nCols, varName, mask, data, target_period, lower, upper, nctimestep, fileName, nocheck, maskout)

Reads forcing input in NetCDF file format.
- subroutine, public [read_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.

15.75.1 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

15.75.2 Function/Subroutine Documentation

15.75.2.1 get_time_vector_and_select()

```
subroutine mo_read_forcing_nc::get_time_vector_and_select (
    type(ncvariable), intent(in) var,
    character(256), intent(in) fname,
    integer(i4), intent(in) inctimestep,
    integer(i4), intent(out) time_start,
    integer(i4), intent(out) time_cnt,
    type(period), intent(in), optional target_period )
```

TODO: add description.

TODO: add description ADDITIONAL INFORMATION get_time_vector_and_select Extract time vector in unit julian hours and get supposed time step in hours

Parameters

in	<i>type(NcVariable) :: var</i>	variable of interest
in	<i>character(256) :: fname</i>	fname of ncfile for error message
in	<i>integer(i4) :: inctimestep</i>	flag for requested time step
out	<i>integer(i4) :: time_start</i>	time_start index of time selection
out	<i>integer(i4) :: time_cnt</i>	time_count of indexes of time selection
in	<i>type(period), optional :: target_period</i>	reference period

Authors

Matthias Zink

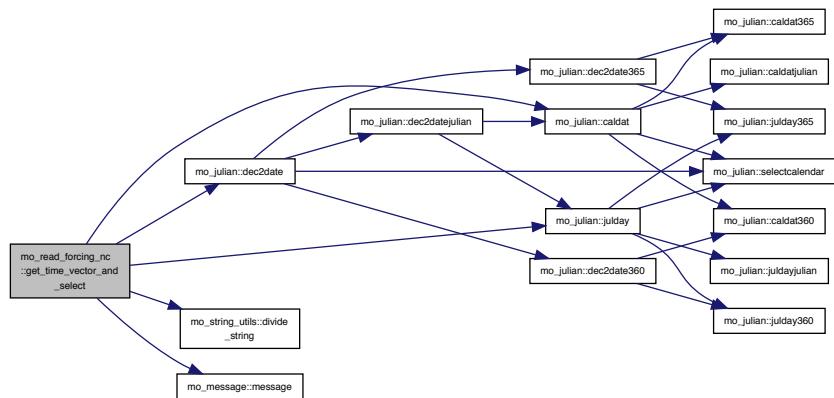
Date

Oct 2012

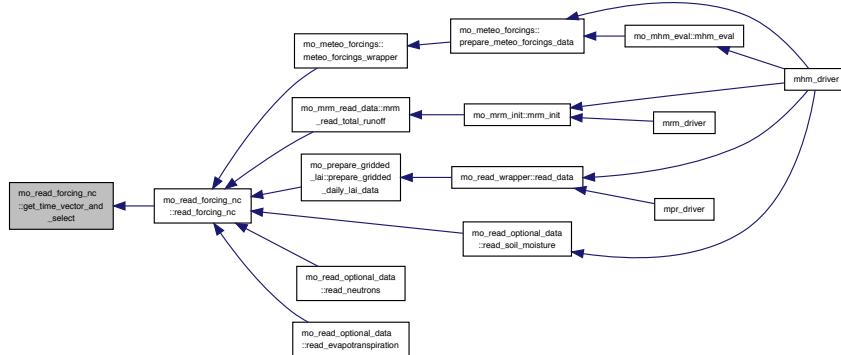
References mo_julian::caldat(), mo_constants::dayhours, mo_constants::daysecs, mo_julian::dec2date(), mo_string_utils::divide_string(), mo_kind::dp, mo_kind::i4, mo_kind::i8, mo_julian::julday(), mo_message::message(), and mo_constants::yeardays.

Referenced by read_forcing_nc().

Here is the call graph for this function:



Here is the caller graph for this function:



15.75.2.2 `read_forcing_nc()`

```
subroutine, public mo_read_forcing_nc::read_forcing_nc (
    character(len = *), intent(in) folder,
    integer(i4), intent(in) nRows,
    integer(i4), intent(in) nCols,
    character(len = *), intent(in) varName,
    logical, dimension(:, :, :), intent(in) mask,
    real(dp), dimension(:, :, :, :), intent(out), allocatable data,
    type(period), intent(in), optional target_period,
    real(dp), intent(in), optional lower,
    real(dp), intent(in), optional upper,
    integer(i4), intent(in), optional nctimestep,
    character(256), intent(in), optional fileName,
    logical, intent(in), optional nocheck,
    logical, dimension(:, :, :, :), intent(out), optional, allocatable maskout )
```

Reads forcing input in NetCDF file format.

Reads netCDF forcing files. First, the dimensions given are cross-checked with header.txt information. Second, the data of the specified period are read from the specified directory. If the optional lower and/or upper bound for the data values is given, the read data are checked for validity. The program is stopped if any value lies out of range. If the optional argument nocheck is true, the data are not checked for coverage with the input mask. Additionally in this case an mask of valid data points can be received from the routine in maskout.

Parameters

in	character(len = *) :: folder	Name of the folder where data are stored
in	integer(i4) :: nRows	Number of datapoints in longitudinal direction
in	integer(i4) :: nCols	Number of datapoints in latitudinal direction
in	character(len = *) :: varName	Name of variable name to read
in	logical, dimension(:, :) :: mask	mask of valid data fields
out	real(dp), dimension(:, :, :) :: data	Data matrix dim_1 = longitude, dim_2 = latitude, 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 :: nc timestep	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 deactivated default = .FALSE. - check is done
out	logical, dimension(:, :, :), optional :: maskout	! mask of valid data points

Authors

Matthias Zink

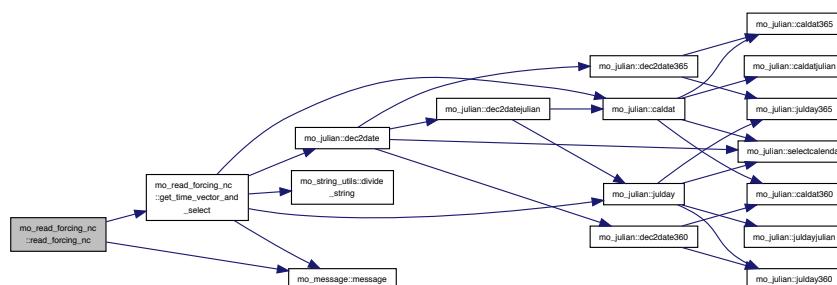
Date

May 2013

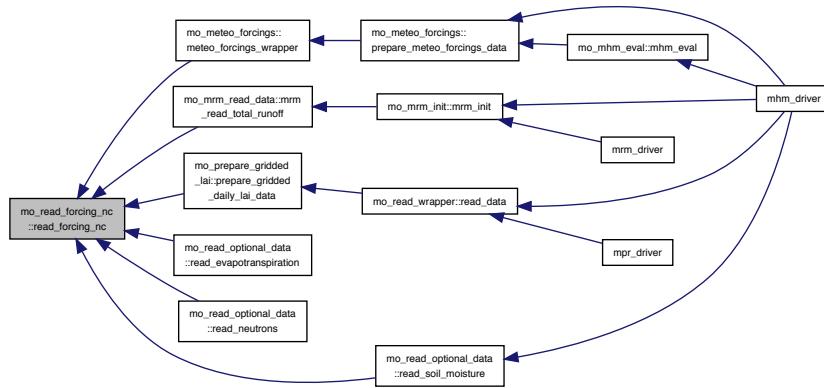
References mo_kind::dp, get_time_vector_and_select(), mo_kind::i4, and mo_message::message().

Referenced by mo_meteo_forcings::meteo_forcings_wrapper(), mo_mrm_read_data::mrm_read_total_runoff(), mo_prepare_gridded_lai::prepare_gridded_daily_lai_data(), mo_read_optional_data::read_evapotranspiration(), mo_read_optional_data::read_neutrons(), and mo_read_optional_data::read_soil_moisture().

Here is the call graph for this function:



Here is the caller graph for this function:



15.75.2.3 read_weights_nc()

```

subroutine, public mo_read_forcing_nc::read_weights_nc (
    character(len = *), intent(in) folder,
    integer(i4), intent(in) nRows,
    integer(i4), intent(in) nCols,
    character(len = *), intent(in) varName,
    real(dp), dimension(:, :, :, :, :), intent(out), allocatable data,
    logical, dimension(:, :, :), intent(in) mask,
    real(dp), intent(in), optional lower,
    real(dp), intent(in), optional upper,
    logical, intent(in), optional nocheck,
    logical, dimension(:, :, :, :, :), intent(out), optional, allocatable maskout,
    character(256), intent(in), optional fileName )

```

Reads weights for meteo forcings input in NetCDF file format.

Reads netCDF weight files. First, the dimensions given are cross-checked with header.txt information. If the optional lower and/or upper bound for the data values is given, the read data are checked for validity. The program is stopped if any value lies out of range. If the optional argument nocheck is true, the data are not checked for coverage with the input mask. Additionally in this case an mask of valid 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 deactivated default = .FALSE. - check is done

Parameters

in	character(256), optional :: fileName	name of variable, defaults to fileName
out	logical, dimension(:, :, :, :), optional :: maskout	! mask of valid data points

Authors

Stephan Thober & Matthias Zink

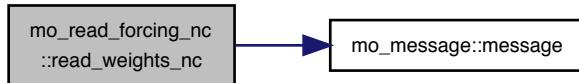
Date

Jan 2017

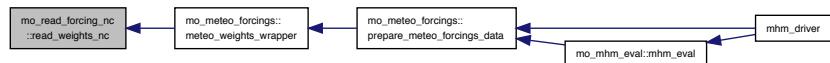
References mo_kind::dp, mo_kind::i4, and mo_message::message().

Referenced by mo_meteo_forcings::meteo_weights_wrapper().

Here is the call graph for this function:



Here is the caller graph for this function:



15.76 mo_read_lation Module Reference

reading latitude and longitude coordinates for each basin

Functions/Subroutines

- subroutine, public `read_lation` (ii, lon_var_name, lat_var_name, level_name, level)
reads latitude and longitude coordinates

15.76.1 Detailed Description

reading latitude and longitude coordinates for each basin

TODO: add description

Authors

Stephan Thober

Date

Nov 2013

15.76.2 Function/Subroutine Documentation

15.76.2.1 read_latlon()

```
subroutine, public mo_read_latlon::read_latlon (
    integer(i4), intent(in) :: ii,
    character(*), intent(in) :: lon_var_name,
    character(*), intent(in) :: lat_var_name,
    character(*), intent(in) :: level_name,
    type(Grid), intent(inout) :: level )
```

reads latitude and longitude coordinates

reads latitude and longitude coordinates from netcdf file for each basin and appends it to the global variables latitude and longitude.

Parameters

in	<i>integer(i4) :: ii</i>	basin indexFile name of the basins must be xxx_latlon.nc, where xxx is the basin id. Variable names in the netcdf file have to be 'lat' for latitude and 'lon' for longitude.
in	<i>character(*) :: lon_var_name</i>	
in	<i>character(*) :: lat_var_name</i>	
in	<i>character(*) :: level_name</i>	
in, out	<i>type(Grid) :: level</i>	

Authors

Stephan Thober

Date

Nov 2013

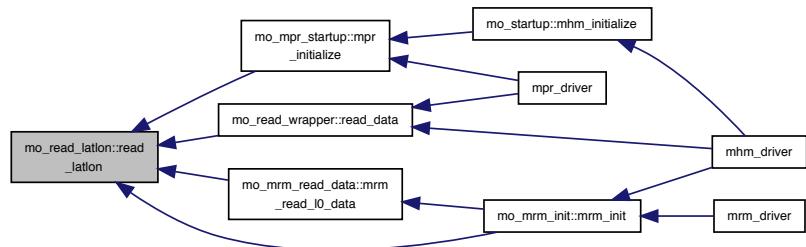
References `mo_common_variables::filelatlon`, and `mo_message::message()`.

Referenced by `mo_mpr_startup::mpr_initialize()`, `mo_mrm_init::mrm_init()`, `mo_mrm_read_data::mrm_read_I0_data()`, and `mo_read_wrapper::read_data()`.

Here is the call graph for this function:



Here is the caller graph for this function:



15.77 mo_read_lut Module Reference

Routines reading lookup tables (lut).

Functions/Subroutines

- subroutine, public `read_geoformation_lut` (filename, fileunit, nGeo, geo_unit, geo_karstic)
Reads LUT containing geological formation information.
- subroutine, public `read_lai_lut` (filename, fileunit, nLAI, LAIIDlist, LAI)
Reads LUT containing LAI information.

15.77.1 Detailed Description

Routines reading lookup tables (lut).

This module contains routines reading various lookup tables (lut). (1) LUT containing gauge information. (2) LUT containing geological formation information. (3) LUT containing LAI class information.

Authors

Juliane Mai, Matthias Zink

Date

Jan 2013

15.77.2 Function/Subroutine Documentation

15.77.2.1 read_geoformation_lut()

```
subroutine, public mo_read_lut::read_geoformation_lut (
    character(len = *), intent(in) filename,
    integer(i4), intent(in) fileunit,
    integer(i4), intent(out) nGeo,
    integer(i4), dimension(:), intent(out), allocatable geo_unit,
    integer(i4), dimension(:), intent(out), allocatable geo_karstic )
```

Reads LUT containing geological formation information.

The LUT needs to have the following header:

```
nGeo_Formations < Number of lines containing data >
GeoParam(i)    ClassUnit      Karstic      Description
```

The subsequent lines contains the geological formation information:

```
<GeoParam(i)> <ClassUnit_i4> <Karstic_i4> <Description_char>
```

All following lines will be discarded while reading. GeoParam is a running index while ClassUnit is the unit of the map containing the geological formations such that it does not necessarily contains subsequent numbers. The parametrization of this unit is part of the namelist mhm_parameter.nml under <geoparameter>.

Parameters

in	character(len = *) :: filename	File name of LUT
in	integer(i4) :: fileunit	Unit to open file
out	integer(i4) :: nGeo	Number of geological formations
out	integer(i4), dimension(:) :: geo_unit	List of id numbers of each geological formations
out	integer(i4), dimension(:) :: geo_karstic	ID of the Karstic formation (0 == does not exist)

Authors

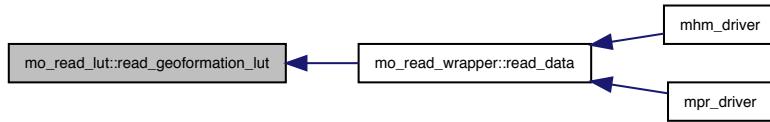
Juliane Mai

Date

Jan 2013

Referenced by `mo_read_wrapper::read_data()`.

Here is the caller graph for this function:



15.77.2.2 `read_lai_lut()`

```
subroutine, public mo_read_lut::read_lai_lut (
    character(len = *), intent(in) filename,
    integer(i4), intent(in) fileunit,
    integer(i4), intent(out) nLAI,
    integer(i4), dimension(:, ), intent(out), allocatable LAIIDlist,
    real(dp), dimension(:, :, ), intent(out), allocatable LAI )
```

Reads LUT containing LAI information.

The LUT needs to have the following header:

```
NoLAIclasses <Number of lines containing data>
Id land-use Jan. Feb. Mar. Apr. May Jun. Jul. Aug. Sep. Oct. Nov. Dec.
```

The subsequent lines contains the lai class information:

```
<ID_i4> <landuse_char> <val_1_dp> <val_2_dp> <val_3_dp> <val_4_dp> ... <val_12_dp>
```

All following lines will be discarded while reading.

Parameters

in	<code>character(len = *) :: filename</code>	File name of LUT
in	<code>integer(i4) :: fileunit</code>	Unit to open file
out	<code>integer(i4) :: nLAI</code>	Number of LAI classes
out	<code>integer(i4), dimension(:) :: LAIIDlist</code>	List of ids of LAI classes
out	<code>real(dp), dimension(:, :,) :: LAI</code>	LAI per class (row) and month (col)

Authors

Juliane Mai

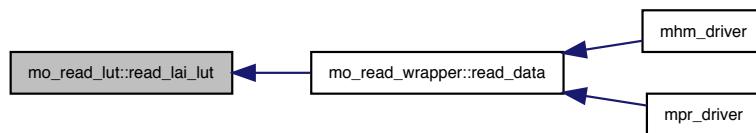
Date

Jan 2013

References mo_kind::i4, and mo_common_constants::yearmonths.

Referenced by mo_read_wrapper::read_data().

Here is the caller graph for this function:



15.78 mo_read_optional_data Module Reference

Read optional data for mHM calibration.

Functions/Subroutines

- subroutine, public **read_soil_moisture** (iBasin)
Read soil moisture data from NetCDF file for calibration.
- subroutine, public **read_basin_avg_tws**
Read basin average TWS timeseries from file, the same way runoff is read.
- subroutine, public **read_neutrons** (iBasin)
Read neutrons data from NetCDF file for calibration.
- subroutine, public **read_evapotranspiration** (iBasin)
Read evapotranspiration data from NetCDF file for calibration.

15.78.1 Detailed Description

Read optional data for mHM calibration.

Data have to be provided in resolution of the hydrology.

Authors

Matthias Zink

Date

Mar 2015

15.78.2 Function/Subroutine Documentation

15.78.2.1 `read_basin_avg_tws()`

```
subroutine, public mo_read_optional_data::read_basin_avg_tws ( )
```

Read basin average TWS timeseries from file, the same way runoff is read.

Read basin average TWS timeseries Allocate global basin_avg_TWS variable that contains the simulated values after the simulation.

Authors

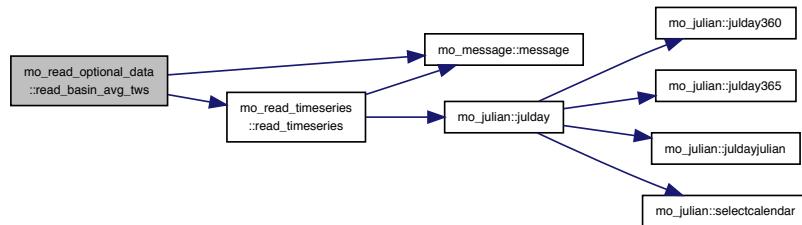
Ondrich Rakovec

Date

Oct 2015

References `mo_global_variables::basin_avg_tws_obs`, `mo_global_variables::basin_avg_tws_sim`, `mo_common_mhm_mrm_variables::evalper`, `mo_message::message()`, `mo_common_variables::nbasins`, `mo_global_variables::nmeasperday_tws`, `mo_common_constants::nodata_dp`, `mo_common_mhm_mrm_variables::ntstepday`, `mo_common_mhm_mrm_variables::opti_function`, `mo_common_mhm_mrm_variables::optimize`, `mo_read_timeseries::read_timeseries()`, `mo_common_mhm_mrm_variables::simper`, and `mo_file::utws`.

Here is the call graph for this function:



15.78.2.2 `read_evapotranspiration()`

```
subroutine, public mo_read_optional_data::read_evapotranspiration (
    integer(i4), intent(in) iBasin )
```

Read evapotranspiration data from NetCDF file for calibration.

This routine reads observed 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	<code>integer(i4) :: iBasin</code>	Basin Id
----	------------------------------------	----------

Authors

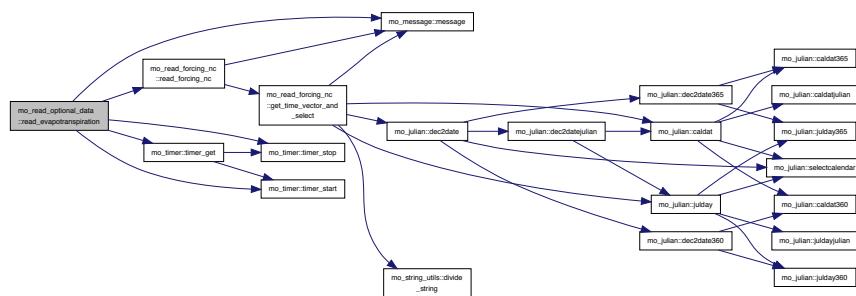
Johannes Brenner

Date

Feb 2017

References `mo_global_variables::direvapotranspiration`, `mo_common_mhm_mrm_variables::evalper`, `mo_global_variables::l1_et`, `mo_global_variables::l1_et_mask`, `mo_common_variables::level1`, `mo_message::message()`, `mo_common_constants::nodata_dp`, `mo_global_variables::ntimesteps_l1_et`, `mo_read_forcing_nc::read_forcing_nc()`, `mo_timer::timer_get()`, `mo_timer::timer_start()`, `mo_timer::timer_stop()`, and `mo_global_variables::timestep_et_input`.

Here is the call graph for this function:



15.78.2.3 read_neutrons()

```
subroutine, public mo_read_optional_data::read_neutrons (
    integer(i4), intent(in) iBasin )
```

Read neutrons data from NetCDF file for calibration.

This routine reads observed 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	<code>integer(i4) :: iBasin</code>	Basin Id
----	------------------------------------	----------

Authors

Martin Schroen

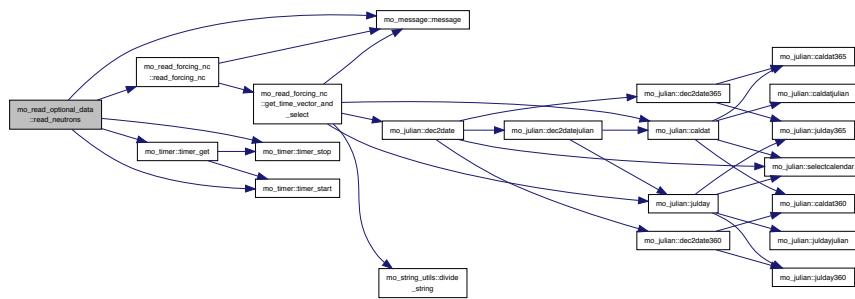
Date

Jul 2015

References `mo_global_variables::dirneutrons`, `mo_common_mhm_mrm_variables::evalper`, `mo_global_variables::l1_neutronsdata`, `mo_global_variables::l1_neutronsdata_mask`, `mo_common_variables::level1`, `mo_message::message()`, `mo_common_constants::nodata_dp`, `mo_global_variables::ntimesteps_l1_neutrons`, `mo_read::read_nc()`, `mo_timer::timer_get()`, `mo_timer::timer_start()`, `mo_timer::timer_stop()`, and `mo_global_variables::timestep_et_input`.

forcing_nc::read_forcing_nc(), mo_timer::timer_get(), mo_timer::timer_start(), mo_timer::timer_stop(), and mo_global_variables::timestep_neutrons_input.

Here is the call graph for this function:



15.78.2.4 read_soil_moisture()

```
subroutine, public mo_read_optional_data::read_soil_moisture (
    integer(i4), intent(in) iBasin )
```

Read soil moisture data from NetCDF file for calibration.

This routine reads observed soil moisture fields which are used for model calibration. The soil moisture file is expected to be called "sm.nc" with a variable "sm" inside. The data are read only for the evaluation period they are intended to be used for calibration. Soil moisture data are only read if one of the corresponding objective functions is chosen.

Parameters

in	integer(i4) :: iBasin	Basin Id
----	-----------------------	----------

Authors

Matthias Zink

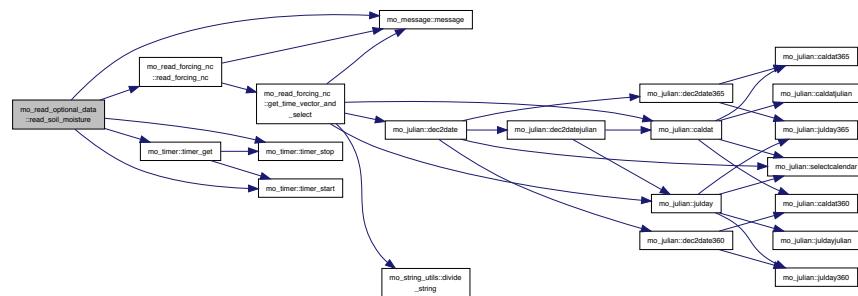
Date

Mar 2015

References `mo_global_variables::dirsoil_moisture`, `mo_common_mhm_mrm_variables::evalper`, `mo_global_variables::l1_sm`, `mo_global_variables::l1_sm_mask`, `mo_common_variables::level1`, `mo_message::message()`, `mo_common_constants::nodata_dp`, `mo_global_variables::ntimesteps_l1_sm`, `mo_read_forcing_nc::read_forcing_nc()`, `mo_timer::timer_get()`, `mo_timer::timer_start()`, `mo_timer::timer_stop()`, and `mo_global_variables::timestep_sm_input`.

Referenced by `mhm_driver()`.

Here is the call graph for this function:



Here is the caller graph for this function:



15.79 mo_read_spatial_data Module Reference

Reads spatial input data.

Data Types

- interface [read_spatial_data_ascii](#)

Reads spatial data files of ASCII format.

Functions/Subroutines

- subroutine [read_spatial_data_ascii_dp](#) (filename, fileunit, header_ncols, header_nrows, header_xllcorner, header_yllcorner, header_cellsize, data, mask)

TODO: add description.

- subroutine `read_spatial_data_ascii_i4` (filename, fileunit, header_ncols, header_nrows, header_xllcorner, header_yllcorner, header_cellsize, data, mask)
TODO: add description.
- subroutine, public `read_header_ascii` (filename, fileunit, header_ncols, header_nrows, header_xllcorner, header_yllcorner, header_cellsize, header_nodata)
Reads header lines of ASCII files.

15.79.1 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

15.79.2 Function/Subroutine Documentation

15.79.2.1 `read_header_ascii()`

```
subroutine, public mo_read_spatial_data::read_header_ascii (
    character(len = *), intent(in) filename,
    integer(i4), intent(in) fileunit,
    integer(i4), intent(out) header_ncols,
    integer(i4), intent(out) header_nrows,
    real(dp), intent(out) header_xllcorner,
    real(dp), intent(out) header_yllcorner,
    real(dp), intent(out) header_cellsize,
    real(dp), intent(out) header_nodata )
```

Reads header lines of ASCII files.

Reads header lines of ASCII files, e.g. dem, aspect, flow direction.

Parameters

in	<code>character(len = *) :: filename</code>	Name of file and its location
in	<code>integer(i4) :: fileunit</code>	File unit for open file
out	<code>integer(i4) :: header_ncols</code>	Reference number of columns
out	<code>integer(i4) :: header_nRows</code>	Reference number of rows
out	<code>real(dp) :: header_xllcorner</code>	Reference lower left corner (x)
out	<code>real(dp) :: header_yllcorner</code>	Reference lower left corner (y)
out	<code>real(dp) :: header_cellsize</code>	Reference cell size [m]
out	<code>real(dp) :: header_nodata</code>	Reference nodata value

Authors

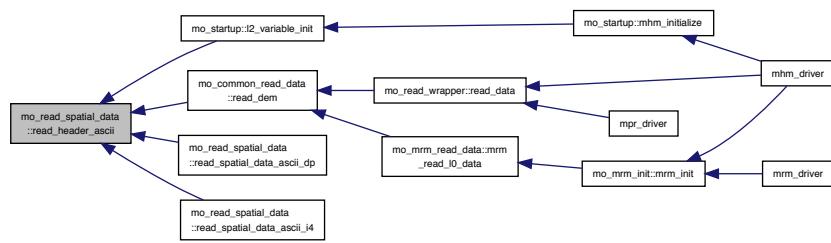
Juliane Mai

Date

Jan 2013

Referenced by mo_startup::l2_variable_init(), mo_common_read_data::read_dem(), read_spatial_data_ascii_dp(), and read_spatial_data_ascii_i4().

Here is the caller graph for this function:



15.79.2.2 read_spatial_data_ascii_dp()

```

subroutine mo_read_spatial_data::read_spatial_data_ascii_dp (
    character(len = *), intent(in) filename,
    integer(i4), intent(in) fileunit,
    integer(i4), intent(in) header_ncols,
    integer(i4), intent(in) header_nrows,
    real(dp), intent(in) header_xllcorner,
    real(dp), intent(in) header_yllcorner,
    real(dp), intent(in) header_cellsize,
    real(dp), dimension(:, :), intent(out), allocatable data,
    logical, dimension(:, :), intent(out), allocatable mask ) [private]

```

TODO: add description.

TODO: add description

Parameters

in	character(len = *) :: filename	filename with location
in	integer(i4) :: fileunit	unit for opening the file
in	integer(i4) :: header_ncols	number of columns of data fields:
in	integer(i4) :: header_nRows	number of rows of data fields:
in	real(dp) :: header_xllcorner	header read in lower left corner
in	real(dp) :: header_yllcorner	header read in lower left corner
in	real(dp) :: header_cellsize	header read in cellsize
out	real(dp), dimension(:, :) :: data	data
out	logical, dimension(:, :) :: mask	mask

Authors

Robert Scheweppe

Date

Jun 2018

References `read_header_ascii()`.

Here is the call graph for this function:



15.79.2.3 `read_spatial_data_ascii_i4()`

```

subroutine mo_read_spatial_data::read_spatial_data_ascii_i4 (
    character(len = *), intent(in) filename,
    integer(i4), intent(in) fileunit,
    integer(i4), intent(in) header_ncols,
    integer(i4), intent(in) header_nrows,
    real(dp), intent(in) header_xllcorner,
    real(dp), intent(in) header_yllcorner,
    real(dp), intent(in) header_cellsize,
    integer(i4), dimension(:, :), intent(out), allocatable data,
    logical, dimension(:, :), intent(out), allocatable mask ) [private]

```

TODO: add description.

TODO: add description

Parameters

in	<code>character(len = *) :: filename</code>	filename with location
in	<code>integer(i4) :: fileunit</code>	unit for opening the file
in	<code>integer(i4) :: header_nCols</code>	number of columns of data fields:
in	<code>integer(i4) :: header_nRows</code>	number of rows of data fields:
in	<code>real(dp) :: header_xllcorner</code>	header read in lower left corner
in	<code>real(dp) :: header_yllcorner</code>	header read in lower left corner
in	<code>real(dp) :: header_cellsize</code>	header read in cellsize
out	<code>integer(i4), dimension(:, :) :: data</code>	data
out	<code>logical, dimension(:, :) :: mask</code>	mask

Authors

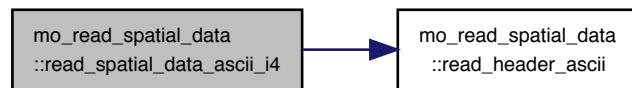
Robert Scheweppe

Date

Jun 2018

References mo_kind::i4, and read_header_ascii().

Here is the call graph for this function:



15.80 mo_read_timeseries Module Reference

Routines to read files containing timeseries data.

Functions/Subroutines

- subroutine, public [read_timeseries](#) (filename, fileunit, periodStart, periodEnd, optimize, opti_function, data, mask, nMeasPerDay)

Reads time series in ASCII format.

15.80.1 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

15.80.2 Function/Subroutine Documentation

15.80.2.1 `read_timeseries()`

```
subroutine, public mo_read_timeseries::read_timeseries (
    character(len = *), intent(in) filename,
    integer(i4), intent(in) fileunit,
    integer(i4), dimension(3), intent(in) periodStart,
    integer(i4), dimension(3), intent(in) periodEnd,
    logical, intent(in) optimize,
    integer(i4), intent(in) opti_function,
    real(dp), dimension(:, ), intent(out), allocatable data,
    logical, dimension(:, ), intent(out), optional, allocatable mask,
    integer(i4), intent(out), optional nMeasPerDay )
```

Reads time series in ASCII format.

Reads time series in ASCII format. Needs specific header lines:

```
<description>
nodata <nodata value>
n <number of measurements per day> measurements per day [1, 1440]
start <YYYY_i4> <MM_i4> <DD_i4> <HH_i4> <MM_i4> (YYYY MM DD HH MM)
end   <YYYY_i4> <MM_i4> <DD_i4> <HH_i4> <MM_i4> (YYYY MM DD HH MM)
```

Line 6 is the first line with data in the following format:

```
<YYYY_i4> <MM_i4> <DD_i4> <HH_i4> <MM_i4> <data_dp>
```

The routine checks for missing data points and if data points are equal distanced. The first data point at each day has to be at HH:MM = 00:00.

Parameters

in	<code>character(len = *) :: filename</code>	File name
in	<code>integer(i4) :: fileunit</code>	Unit to open file
in	<code>integer(i4), dimension(3) :: periodStart</code>	Start day of reading (YYYY,MM,DD)
in	<code>integer(i4), dimension(3) :: periodEnd</code>	End day of reading (YYYY,MM,DD)
in	<code>logical :: optimize</code>	optimization flag
in	<code>integer(i4) :: opti_function</code>	
out	<code>real(dp), dimension(:,) :: data</code>	Data vector
out	<code>logical, dimension(:,), optional :: mask</code>	Mask for nodata values in data
out	<code>integer(i4), optional :: nMeasPerDay</code>	Number of data points per day

Authors

Matthias Zink, Juliane Mai

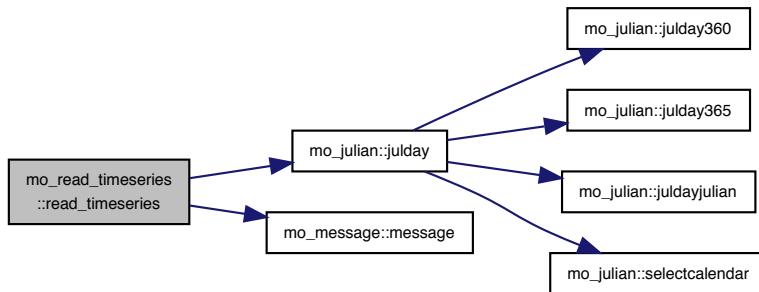
Date

Jan 2013

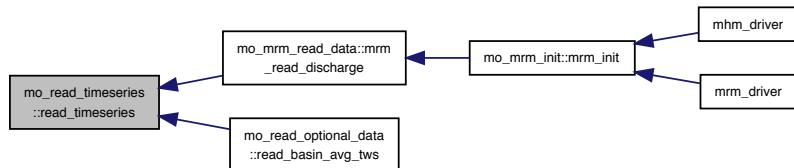
References `mo_julian::julday()`, and `mo_message::message()`.

Referenced by `mo_mrm_read_data::mrm_read_discharge()`, and `mo_read_optional_data::read_basin_avg_tws()`.

Here is the call graph for this function:



Here is the caller graph for this function:



15.81 mo_read_wrapper Module Reference

Wrapper for all reading routines.

Functions/Subroutines

- subroutine, public `read_data` (LAIPer)

Reads data.
- subroutine `check_consistency_lut_map` (data, lookuptable, filename, unique_values)

Checks if classes in input maps appear in look up tables.

15.81.1 Detailed Description

Wrapper for all reading routines.

This module is to wrap up all reading routines. The general written reading routines are used to store now the read data into global variables.

Authors

Juliane Mai, Matthias Zink

Date

Jan 2013

15.81.2 Function/Subroutine Documentation

15.81.2.1 `check_consistency_lut_map()`

```
subroutine mo_read_wrapper::check_consistency_lut_map (
    integer(i4), dimension(:), intent(in) data,
    integer(i4), dimension(:), intent(in) lookuptable,
    character(*), intent(in) filename,
    integer(i4), dimension(:), intent(out), optional, allocatable unique_values )
```

Checks if classes in input maps appear in look up tables.

Determines whether a class appearing in the morphological input is occurring in the respective look up table. mHM breaks if inconsistencies are discovered.

Parameters

in	<i>integer(i4), dimension(:) :: data</i>	map of study domain
in	<i>integer(i4), dimension(:) :: lookuptable</i>	look up table corresponding to map
in	<i>character(*) :: filename</i>	name of the lut file - ERRRR warn
out	<i>integer(i4), dimension(:), optional :: unique_values</i>	array of unique values in dataone

Authors

Matthias Zink

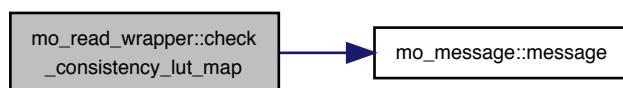
Date

Nov 2016

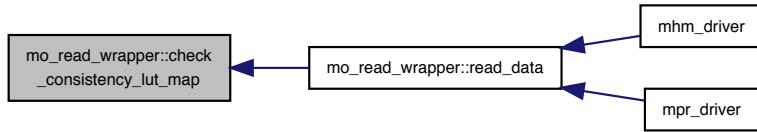
References `mo_message::message()`.

Referenced by `read_data()`.

Here is the call graph for this function:



Here is the caller graph for this function:



15.81.2.2 read_data()

```
subroutine, public mo_read_wrapper::read_data (
    type(period), dimension(:), intent(in), optional LAIPer )
```

Reads data.

The namelists are already read by read_config call. All LUTs are read from their respective directory and information within those files are shared across all basins to be modeled.

Parameters

in	type(period), dimension(:), optional :: LAIPer	
----	--	--

Authors

Juliane Mai & Matthias Zink

Date

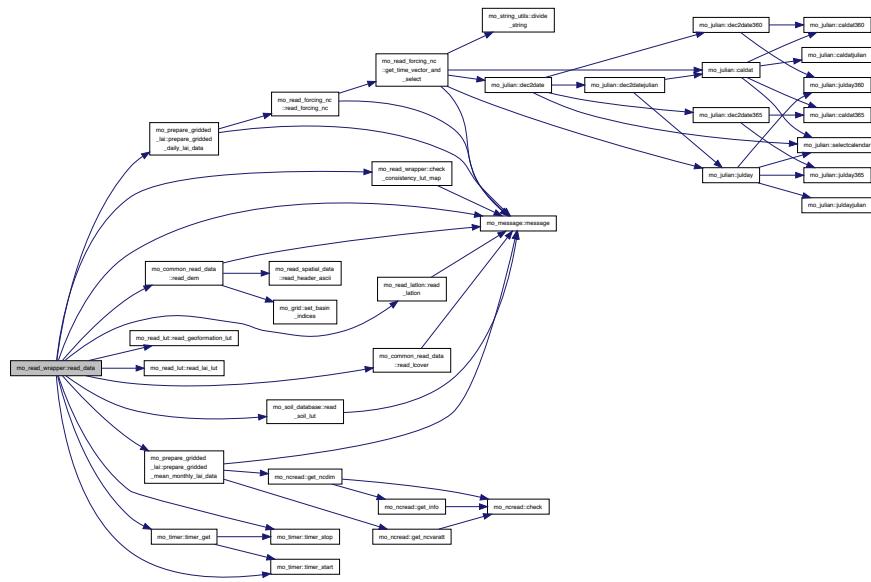
Feb 2013

by default; when iFlag_soilDB = 0

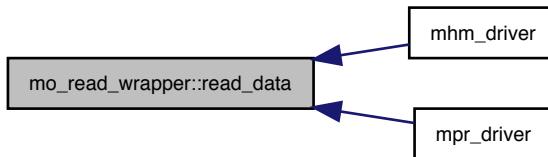
References check_consistency_lut_map(), mo_common_variables::dircommonfiles, mo_common_variables:::dirmorpho, mo_mpr_file::file_aspect, mo_mpr_file::file_geolut, mo_mpr_file::file_hydrogeoclass, mo_mpr_file::file_laiclass, mo_mpr_file::file_lailut, mo_mpr_file::file_slope, mo_mpr_file::file_soil_database, mo_mpr_file::file_soil_database_1, mo_mpr_file::file_soilclass, mo_mpr_global_variables::geounitkar, mo_mpr_global_variables::geounitlist, mo_common_variables::global_parameters, mo_mpr_global_variables::iflag_soildb, mo_mpr_global_variables::i0_asp, mo_common_variables::i0_basin, mo_mpr_global_variables::i0_geounit, mo_mpr_global_variables::i0_gridded_lai, mo_mpr_global_variables::i0_slope, mo_mpr_global_variables::i0_soilid, mo_mpr_global_variables::lailut, mo_mpr_global_variables::laiunitlist, mo_common_variables::level0, mo_message::message(), mo_common_variables::nbasins, mo_mpr_global_variables::ngeounits, mo_mpr_global_variables::nlai, mo_mpr_global_variables::nlaiclass, mo_common_constants::nodata_dp, mo_common_constants::nodata_i4, mo_mpr_global_variables::nsoilhorizons_mhm, mo_prepare_gridded_lai::prepare_gridded_daily_lai_data(), mo_prepare_gridded_lai::prepare_gridded_mean_monthly_lai_data(), mo_common_variables::processmatrix, mo_common_read_data::read_dem(), mo_read_lut::read_geoformation_lut(), mo_read_lut::read_lai_lut(), mo_read_latlon::read_latlon(), mo_common_read_data::read_lcover(), mo_soil_database::read_soil_lut(), mo_mpr_global_variables::soildb, mo_timer::timer_get(), mo_timer::timer_start(), mo_timer::timer_stop(), mo_mpr_global_variables::timestep_lai_input, mo_mpr_file::uaspect, mo_mpr_file::ugeolut, mo_mpr_file::uhydrogeoclass, mo_mpr_file::ulaiclass, mo_mpr_file::ulailut, mo_mpr_file::uslope, mo_mpr_file::usoilclass, and mo_common_constants::yearmonths_i4.

Referenced by `mhm_driver()`, and `mpr_driver()`.

Here is the call graph for this function:



Here is the caller graph for this function:



15.82 mo restart Module Reference

reading and writing states, fluxes and configuration for restart of mHM.

Data Types

- interface `unpack_field_and_write`
TODO: add description.

Functions/Subroutines

- subroutine, public `write_restart_files` (OutPath)
write restart files for each basin

- subroutine, public [read_restart_states](#) (iBasin, InPath)
reads fluxes and state variables from file
- subroutine [unpack_field_and_write_1d_i4](#) (nc, var_name, var_dims, fill_value, data, mask, var_long_name)
- subroutine [unpack_field_and_write_1d_dp](#) (nc, var_name, var_dims, fill_value, data, mask, var_long_name)
- subroutine [unpack_field_and_write_2d_dp](#) (nc, var_name, var_dims, fill_value, data, mask, var_long_name)
- subroutine [unpack_field_and_write_3d_dp](#) (nc, var_name, var_dims, fill_value, data, mask, var_long_name)

15.82.1 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

15.82.2 Function/Subroutine Documentation

15.82.2.1 [read_restart_states\(\)](#)

```
subroutine, public mo_restart::read_restart_states (
    integer(i4), intent(in) iBasin,
    character(256), intent(in) InPath )
```

reads fluxes and state variables from file

read fluxes and state variables from given restart directory and initialises all state variables that are initialized in the subroutine initialise, contained in module [mo_startup](#).

Parameters

in	<i>integer(i4) :: iBasin</i>	number of basin
in	<i>character(256) :: InPath</i>	Input Path including trailing slash

Authors

Stephan Thober

Date

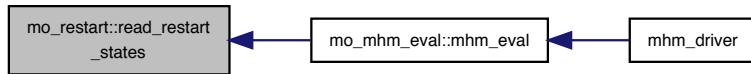
Apr 2013

References `mo_kind::dp`, `mo_kind::i4`, `mo_mpr_global_variables::l1_aeroresist`, `mo_global_variables::l1_aetcanopy`, `mo_global_variables::l1_aetsealed`, `mo_global_variables::l1_aetsoil`, `mo_mpr_global_variables::l1_alpha`, `mo_global_variables::l1_baseflow`, `mo_mpr_global_variables::l1_degday`, `mo_mpr_global_variables::l1`

`_deggdayinc, mo_mpr_global_variables::l1_deggdaymax, mo_mpr_global_variables::l1_deggdaynopre, mo_mpr_global_variables::l1_fasp, mo_global_variables::l1_fastrunoff, mo_mpr_global_variables::l1_froots, mo_mpr_global_variables::l1_fsealed, mo_mpr_global_variables::l1_harsamcoeff, mo_global_variables::l1_infilsoil, mo_global_variables::l1_inter, mo_mpr_global_variables::l1_jarvis_thresh_c1, mo_mpr_global_variables::l1_karstloss, mo_mpr_global_variables::l1_kbaseflow, mo_mpr_global_variables::l1_kfastflow, mo_mpr_global_variables::l1_kperco, mo_mpr_global_variables::l1_kslowflow, mo_mpr_global_variables::l1_maxinter, mo_global_variables::l1_melt, mo_global_variables::l1_percol, mo_mpr_global_variables::l1_petlaicfactor, mo_global_variables::l1_preeffect, mo_mpr_global_variables::l1_prietaryalpha, mo_global_variables::l1_rain, mo_global_variables::l1_runoffseal, mo_global_variables::l1_satstw, mo_mpr_global_variables::l1_sealedthresh, mo_global_variables::l1_sealstw, mo_global_variables::l1_slowrunoff, mo_global_variables::l1_snow, mo_global_variables::l1_snowpack, mo_global_variables::l1_soilmoist, mo_mpr_global_variables::l1_soilmoistexp, mo_mpr_global_variables::l1_soilmoistfc, mo_mpr_global_variables::l1_soilmoistsat, mo_mpr_global_variables::l1_surfresist, mo_mpr_global_variables::l1_tempthresh, mo_global_variables::l1_throughfall, mo_global_variables::l1_total_runoff, mo_global_variables::l1_unsatstw, mo_mpr_global_variables::l1_unsatthresh, mo_mpr_global_variables::l1_wiltingpoint, mo_common_variables::lc_year_end, mo_common_variables::lc_year_start, mo_common_variables::level1, mo_mpr_global_variables::nlai, mo_common_variables::nlcoverscene, mo_mpr_global_variables::nsoilhorizons_mhm, and mo_common_variables::processmatrix.`

Referenced by `mo_mhm_eval::mhm_eval()`.

Here is the caller graph for this function:



15.82.2.2 `unpack_field_and_write_1d_dp()`

```

subroutine mo_restart::unpack_field_and_write_1d_dp (
    type(ncdataset), intent(inout) nc,
    character(*), intent(in) var_name,
    type(ncdimension), dimension(:), intent(in) var_dims,
    real(dp), intent(in) fill_value,
    real(dp), dimension(:), intent(in) data,
    logical, dimension(:, :), intent(in) mask,
    character(*), intent(in), optional var_long_name )

```

References `mo_kind::dp`.

15.82.2.3 `unpack_field_and_write_1d_i4()`

```

subroutine mo_restart::unpack_field_and_write_1d_i4 (
    type(ncdataset), intent(inout) nc,
    character(*), intent(in) var_name,
    type(ncdimension), dimension(:), intent(in) var_dims,
    integer(i4), intent(in) fill_value,
    integer(i4), dimension(:), intent(in) data,
    logical, dimension(:, :), intent(in) mask,
    character(*), intent(in), optional var_long_name )

```

References mo_kind::i4.

15.82.2.4 unpack_field_and_write_2d_dp()

```
subroutine mo_restart::unpack_field_and_write_2d_dp (
    type(ncdataset), intent(inout) nc,
    character(*), intent(in) var_name,
    type(ncdimension), dimension(:), intent(in) var_dims,
    real(dp), intent(in) fill_value,
    real(dp), dimension(:, :, :), intent(in) data,
    logical, dimension(:, :, :), intent(in) mask,
    character(*), intent(in), optional var_long_name )
```

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

15.82.2.5 unpack_field_and_write_3d_dp()

```
subroutine mo_restart::unpack_field_and_write_3d_dp (
    type(ncdataset), intent(inout) nc,
    character(*), intent(in) var_name,
    type(ncdimension), dimension(:), intent(in) var_dims,
    real(dp), intent(in) fill_value,
    real(dp), dimension(:, :, :, :), intent(in) data,
    logical, dimension(:, :, :, :), intent(in) mask,
    character(*), intent(in), optional var_long_name )
```

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

15.82.2.6 write_restart_files()

```
subroutine, public mo_restart::write_restart_files (
    character(256), dimension(:), intent(in) OutPath )
```

write restart files for each basin

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

Parameters

in	character(256), dimension(:) :: OutPath	Output Path for each basin
----	---	----------------------------

Authors

Stephan Thober

Date

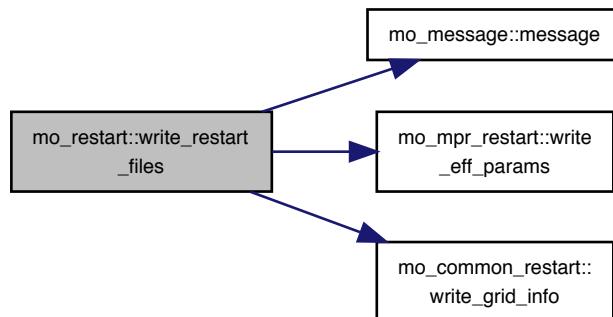
Jun 2014

References mo_kind::dp, mo_kind::i4, mo_global_variables::l1_aetcanopy, mo_global_variables::l1_aetsealed, mo_global_variables::l1_aetsoil, mo_global_variables::l1_baseflow, mo_global_variables::l1_fastrunoff, mo←

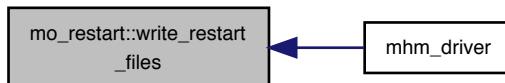
global_variables::l1_infilsoil, mo_global_variables::l1_inter, mo_global_variables::l1_melt, mo_global_variables::l1_percol, mo_global_variables::l1_preeffect, mo_global_variables::l1_rain, mo_global_variables::l1_runoffseal, mo_global_variables::l1_satstw, mo_global_variables::l1_sealstw, mo_global_variables::l1_slowrunoff, mo_global_variables::l1_snow, mo_global_variables::l1_snowpack, mo_global_variables::l1_soilmoist, mo_global_variables::l1_throughfall, mo_global_variables::l1_total_runoff, mo_global_variables::l1_unsatstw, mo_common_variables::level1, mo_message::message(), mo_mpr_global_variables::nlai, mo_common_variables::nlcoverscene, mo_common_constants::nodata_dp, mo_mpr_global_variables::nsoilhorizons_mhm, mo_mpr_restart::write_eff_params(), and mo_common_restart::write_grid_info().

Referenced by mhm_driver().

Here is the call graph for this function:



Here is the caller graph for this function:



15.83 mo_runoff Module Reference

Runoff generation for the unsaturated zone, saturated zone (or groundwater zone), and runoff accumulation.

Functions/Subroutines

- subroutine, public [runoff_unsat_zone](#) (k1, kp, k0, alpha, karst_loss, pefec_soil, unsat_thresh, sat_storage, unsat_storage, slow_interflow, fast_interflow, perc)

Runoff generation for the saturated zone.

- subroutine, public [runoff_sat_zone](#) (k2, sat_storage, baseflow)

Runoff generation for the saturated zone.

- subroutine, public `l1_total_runoff` (`fSealed_area_fraction`, `fast_interflow`, `slow_interflow`, `baseflow`, `direct_runoff`, `total_runoff`)

total runoff accumulation at level 1

15.83.1 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

15.83.2 Function/Subroutine Documentation

15.83.2.1 l1_total_runoff()

```
subroutine, public mo_runoff::l1_total_runoff (
    real(dp), intent(in) fSealed_area_fraction,
    real(dp), intent(in) fast_interflow,
    real(dp), intent(in) slow_interflow,
    real(dp), intent(in) baseflow,
    real(dp), intent(in) direct_runoff,
    real(dp), intent(out) total_runoff )
```

total runoff accumulation at level 1

Accumulates runoff.

$$q_T = (q_0 + q_1 + q_2) * (1 - fSealed) + q_D * fSealed$$

, where `fSealed` is the fraction of sealed area.

Parameters

in	<code>REAL(dp) :: fSealed_area_fraction</code>	sealed area fraction [1]
in	<code>REAL(dp) :: fast_interflow</code>	q_0 Fast runoff component [mm tst-1]
in	<code>REAL(dp) :: slow_interflow</code>	q_1 Slow runoff component [mm tst-1]
in	<code>REAL(dp) :: baseflow</code>	q_2 Baseflow [mm ts-1]
in	<code>REAL(dp) :: direct_runoff</code>	q_D Direct runoff from impervious areas [mm tst-1]
out	<code>REAL(dp) :: total_runoff</code>	q_T Generated runoff [mm tst-1]

Authors

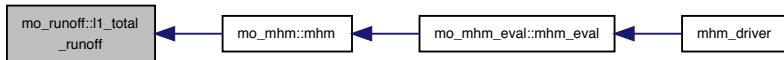
Vladyslav Prykhodko

Date

Dec 2012

Referenced by `mo_mhm::mhm()`.

Here is the caller graph for this function:

**15.83.2.2 runoff_sat_zone()**

```

subroutine, public mo_runoff::runoff_sat_zone (
    real(dp), intent(in) k2,
    real(dp), intent(inout) sat_storage,
    real(dp), intent(out) baseflow )
  
```

Runoff generation for the saturated zone.

Calculates the runoff generation for the saturated zone. If the level of the ground water reservoir is zero, then the baseflow is also zero. If the level of the ground water reservoir is greater than zero, then the baseflow is equal to baseflow recession coefficient times the level of the ground water reservoir, which will be then reduced by the value of baseflow.

Parameters

in	<i>REAL(dp) :: k2</i>	Baseflow recession coefficient [d-1]
in,out	<i>REAL(dp) :: sat_storage</i>	Groundwater storage [mm]
out	<i>REAL(dp) :: baseflow</i>	Baseflow [mm d-1]

Authors

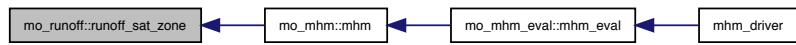
Vladyslav Prykhodko

Date

Dec 2012

Referenced by `mo_mhm::mhm()`.

Here is the caller graph for this function:



15.83.2.3 runoff_unsat_zone()

```
subroutine, public mo_runoff::runoff_unsat_zone (
    real(dp), intent(in) k1,
    real(dp), intent(in) kp,
    real(dp), intent(in) k0,
    real(dp), intent(in) alpha,
    real(dp), intent(in) karst_loss,
    real(dp), intent(in) pefec_soil,
    real(dp), intent(in) unsat_thresh,
    real(dp), intent(inout) sat_storage,
    real(dp), intent(inout) unsat_storage,
    real(dp), intent(out) slow_interflow,
    real(dp), intent(out) fast_interflow,
    real(dp), intent(out) perc )
```

Runoff generation for the saturated zone.

Calculates the runoff generation for the unsaturated zone. Calculates percolation, interflow and baseflow. Updates upper soil and groundwater storages.

Parameters

in	REAL(dp) :: k1	Recession coefficient of the upper reservoir,lower outlet [d-1]
in	REAL(dp) :: kp	Percolation coefficient [d-1]
in	REAL(dp) :: k0	Recession coefficient of the upper reservoir, upper outlet [d-1]
in	REAL(dp) :: alpha	Exponent for the upper reservoir [-]
in	REAL(dp) :: karst_loss	Karstic percolation loss [-]
in	REAL(dp) :: pefec_soil	Input to the soil layer [mm]
in	REAL(dp) :: unsat_thresh	Threshold water depth in upper reservoir(for Runoff contribution) [mm]
in,out	REAL(dp) :: sat_storage	Groundwater storage [mm]
in,out	REAL(dp) :: unsat_storage	Upper soil storage [mm]
out	REAL(dp) :: slow_interflow	Slow runoff component [mm d-1]
out	REAL(dp) :: fast_interflow	Fast runoff component [mm d-1]
out	REAL(dp) :: perc	Percolation [mm d-1]

Authors

Vladyslav Prykhodko

Date

Dec 2012

References mo_common_constants::eps_dp.

Referenced by mo_mhm::mhm().

Here is the caller graph for this function:



15.84 mo_sce Module Reference

Shuffled Complex Evolution optimization algorithm.

Functions/Subroutines

- real(dp) function, dimension(size(pini, 1)), public **sce** (eval, functn, pini, prange, mymaxn, mymaxit, mykstop, mypcento, mypeps, myseed, myngs, mynpg, mynps, mynspl, mymings, myiniflg, myprint, mymask, myalpha, mybeta, tmp_file, popul_file, popul_file_append, parallel, restart, restart_file, bestf, neval, history)

Shuffled Complex Evolution (SCE) algorithm for global optimization.

- subroutine **parstt** (x, bound, peps, mask, xnstd, gnrng, ipcnvg)
- subroutine **comp** (ngs2, npg, a, af, b, bf)
- subroutine **sort_matrix** (rb, ra)
- subroutine **chkcst** (x, bl, bu, mask, ibound)
- subroutine **getpnt** (idist, bl, bu, std, xi, mask, save_state, x)
- subroutine **cce** (s, sf, bl, bu, maskpara, xnstd, icall, maxn, maxit, save_state_gauss, functn, eval, alpha, beta, history, idot)

15.84.1 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

15.84.2 Function/Subroutine Documentation

15.84.2.1 cce()

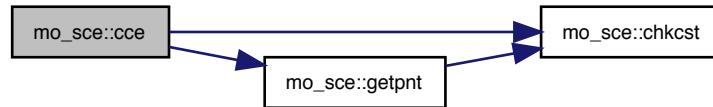
```
subroutine mo_sce::cce (
    real(dp), dimension(:, :, intent(inout) s,
    real(dp), dimension(:, intent(inout) sf,
    real(dp), dimension(:, intent(in) bl,
    real(dp), dimension(:, intent(in) bu,
    logical, dimension(:, intent(in) maskpara,
    real(dp), dimension(:, intent(in) xnstd,
    integer(i8), intent(inout) icall,
    integer(i8), intent(in) maxn,
    logical, intent(in) maxit,
    integer(i8), dimension(n_save_state), intent(inout) save_state_gauss,
    procedure(objective_interface), intent(in), pointer functn,
    procedure(eval_interface), intent(in), pointer eval,
    real(dp), intent(in) alpha,
    real(dp), intent(in) beta,
```

```
real(dp), dimension(maxn), intent(inout) history,
logical, intent(in) idot )
```

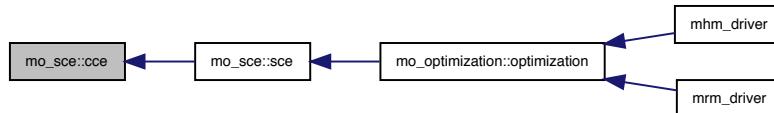
References chkcst(), mo_kind::dp, getpnt(), mo_kind::i4, mo_kind::i8, and mo_xor4096::n_save_state.

Referenced by sce().

Here is the call graph for this function:



Here is the caller graph for this function:



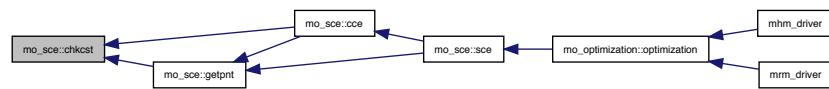
15.84.2.2 chkcst()

```
subroutine mo_sce::chkcst (
    real(dp), dimension(:), intent(in) x,
    real(dp), dimension(:), intent(in) bl,
    real(dp), dimension(:), intent(in) bu,
    logical, dimension(:), intent(in) mask,
    integer(i4), intent(out) ibound )
```

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

Referenced by cce(), and getpnt().

Here is the caller graph for this function:



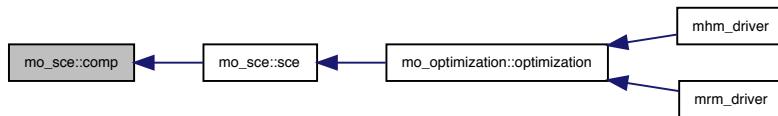
15.84.2.3 comp()

```
subroutine mo_sce::comp (
    integer(i4), intent(in) ngs2,
    integer(i4), intent(in) npg,
    real(dp), dimension(:, :), intent(inout) a,
    real(dp), dimension(:, :), intent(inout) af,
    real(dp), dimension(size(a, 1), size(a, 2)), intent(out) b,
    real(dp), dimension(size(af)), intent(out) bf )
```

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

Referenced by sce().

Here is the caller graph for this function:



15.84.2.4 getpnt()

```
subroutine mo_sce::getpnt (
    integer(i4), intent(in) idist,
    real(dp), dimension(:, :), intent(in) bl,
    real(dp), dimension(:, :), intent(in) bu,
    real(dp), dimension(:, :), intent(in) std,
    real(dp), dimension(:, :), intent(in) xi,
    logical, dimension(:, :), intent(in) mask,
    integer(i8), dimension(n_save_state), intent(inout) save_state,
    real(dp), dimension(size(xi, 1)), intent(out) x )
```

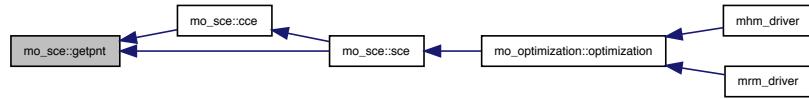
References chkcst(), mo_kind::dp, mo_kind::i4, mo_kind::i8, and mo_xor4096::n_save_state.

Referenced by cce(), and sce().

Here is the call graph for this function:



Here is the caller graph for this function:



15.84.2.5 parstt()

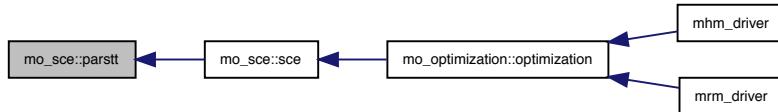
```

subroutine mo_sce::parstt (
    real(dp), dimension(:, :, ), intent(in) x,
    real(dp), dimension(:, ), intent(in) bound,
    real(dp), intent(in) peps,
    logical, dimension(:, ), intent(in) mask,
    real(dp), dimension(size(bound, 1)), intent(out) xnstd,
    real(dp), intent(out) gnrng,
    integer(i4), intent(out) ipcnavg )  [private]
  
```

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

Referenced by sce().

Here is the caller graph for this function:



15.84.2.6 sce()

```

real(dp) function, dimension(size(pini, 1)), public mo_sce::sce (
    procedure(eval_interface), intent(in), pointer eval,
    procedure(objective_interface), intent(in), pointer functn,
    real(dp), dimension(:, ), intent(in) pini,
    real(dp), dimension(:, :, ), intent(in) prange,
    integer(i8), intent(in), optional mymaxn,
    logical, intent(in), optional mymaxit,
    integer(i4), intent(in), optional mykstop,
    real(dp), intent(in), optional mypcnto,
    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	<i>real(dp) :: functn(p)</i>	Function on which to search the optimum
in	<i>real(dp) :: pini(:)</i>	initial value of decision variables
in	<i>real(dp) :: prange(size(pini),2)</i>	Min/max range of decision variables
in	<i>integer(i8), optional :: mymaxn</i>	max no. of trials allowed before optimization is terminated DEFAULT: 1000_i8

Parameters

in	<i>logical, optional :: mymaxit</i>	maximization (.true.) or minimization (.false.) of function DEFAULT: false
in	<i>integer(i4), optional :: mykstop</i>	number of shuffling loops in which the criterion value must change by given percentage before optimiz. is terminated DEFAULT: 10_i4
in	<i>real(dp), optional :: mypcnto</i>	percentage by which the criterion value must change in given number of shuffling loops DEFAULT: 0.0001_dp
in	<i>real(dp), optional :: mypeps</i>	optimization is terminated if volume of complex has converged to given percentage of feasible space DEFAULT: 0.001_dp
in	<i>integer(i8), optional :: myseed</i>	initial random seed DEFAULT: get_timeseed
in	<i>integer(i4), optional :: myngs</i>	number of complexes in the initial population DEFAULT: 2_i4
in	<i>integer(i4), optional :: mynpg</i>	number of points in each complex DEFAULT: 2*n+1
in	<i>integer(i4), optional :: mynps</i>	number of points in a sub-complex DEFAULT: n+1
in	<i>integer(i4), optional :: mynspl</i>	number of evolution steps allowed for each complex before complex shuffling DEFAULT: 2*n+1
in	<i>integer(i4), optional :: mymings</i>	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	<i>integer(i4), optional :: myiniflg</i>	flag on whether to include the initial point in population 0, not included 1, included (DEFAULT)
in	<i>integer(i4), optional :: myprint</i>	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	<i>logical, optional :: mymask(size(pini))</i>	parameter included in optimization (true) or discarded (false) DEFAULT: .true.
in	<i>real(dp), optional :: myalpha</i>	parameter for reflection of points in complex DEFAULT: 0.8_dp
in	<i>real(dp), optional :: mybeta</i>	parameter for contraction of points in complex DEFAULT: 0.45_dp
in	<i>character(len=*), optional :: tmp_file</i>	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	<i>character(len=*)</i> , optional :: <i>popul_file</i>	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	<i>logical</i> , optional :: <i>popul_file_append</i>	if true, append to existing population file (default: false)
in	<i>logical</i> , optional :: <i>parallel</i>	sce runs in parallel (true) or not (false) parallel sce should only be used if model/ objective is not parallel DEFAULT: .false.
in	<i>logical</i> , optional :: <i>restart</i>	if .true.: restart former sce run from <i>restart_file</i>
in	<i>character(len=*)</i> , optional :: <i>restart_file</i>	file name for read/write of restart file (default: mo_sce.restart)
out	<i>real(dp)</i> , optional :: <i>bestf</i>	the best value of the function.
out	<i>integer(i8)</i> , optional :: <i>neval</i>	number of function evaluations needed.
out	<i>real(dp)</i> , optional, allocatable :: <i>history(:)</i>	the history of best function values, <i>history(neval)=bestf</i>

Returns

real(dp) :: *bestx(size(pini))* — The parameters of the point which is estimated to minimize/maximize the function.

Note

Maximal number of parameters is 1000.

SCE is OpenMP enabled on the loop over the complexes.

OMP_NUM_THREADS > 1 does not give reproducible results even when seeded!

Author

Juliane Mai

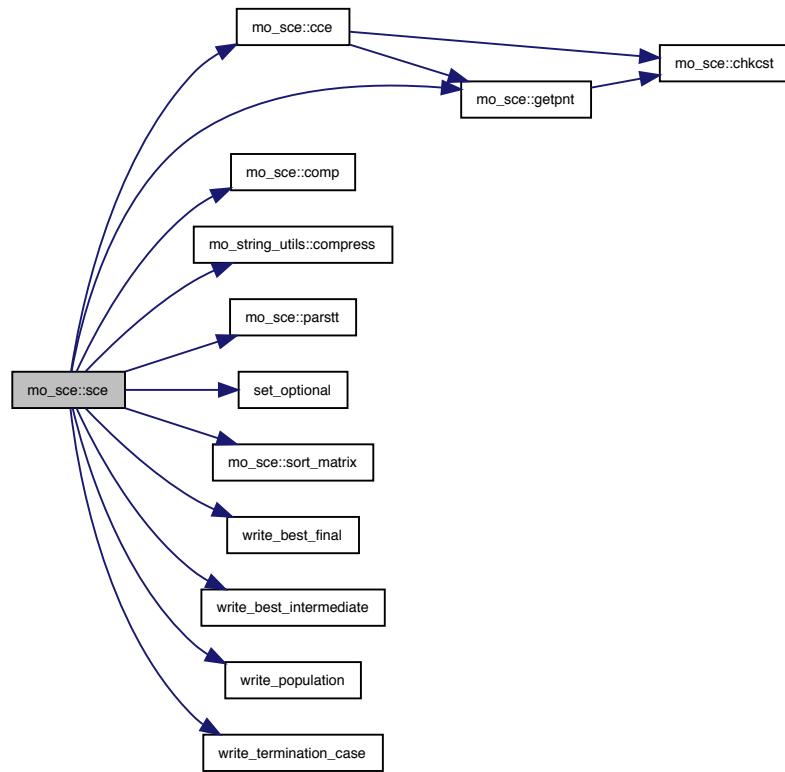
Date

Feb 2013

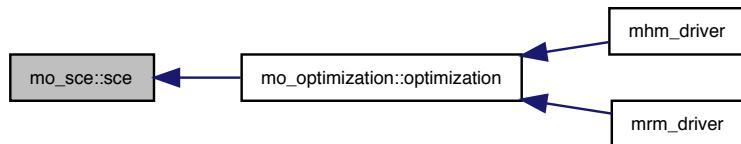
References *cce()*, *comp()*, *mo_string_utils::compress()*, *mo_kind::dp*, *getpnt()*, *mo_kind::i4*, *mo_kind::i8*, *mo_xor4096::n_save_state*, *parstt()*, *set_optional()*, *sort_matrix()*, *write_best_final()*, *write_best_intermediate()*, *write_population()*, and *write_termination_case()*.

Referenced by *mo_optimization::optimization()*.

Here is the call graph for this function:



Here is the caller graph for this function:



15.84.2.7 sort_matrix()

```

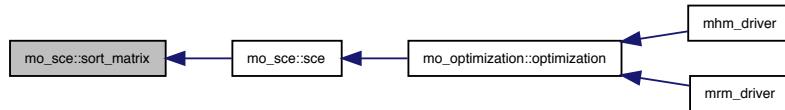
subroutine mo_sce::sort_matrix (
    real(dp), dimension(:, :), intent(inout) rb,
    real(dp), dimension(:, :), intent(inout) ra )

```

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

Referenced by `sce()`.

Here is the caller graph for this function:



15.85 mo_set_ncdf_outputs Module Reference

Defines the structure of the netCDF to write the output in.

Functions/Subroutines

- subroutine, public [set_ncdf](#) (NoNetcdfVars, nrows, ncols)
Runs mhm with a specific parameter set and returns required variables, e.g. runoff.

15.85.1 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

15.85.2 Function/Subroutine Documentation

15.85.2.1 [set_ncdf\(\)](#)

```
subroutine, public mo_set_ncdf_outputs::set_ncdf (
    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	<i>integer(i4) :: NoNetcdfVars</i>	
in	<i>integer(i4) :: nrows</i>	
in	<i>integer(i4) :: ncols</i>	

Authors

Matthias Zink

Date

Apr 2013

References mo_ncwrite::dnc, mo_ncwrite::ndims, mo_ncwrite::nvars, and mo_ncwrite::v.

15.86 mo_snow_accum_melt Module Reference

Snow melting and accumulation.

Functions/Subroutines

- subroutine, public `snow_accum_melt` (deg_day_incr, deg_day_max, deg_day_noprec, prec, temperature, temperature_thresh, thrfall, snow_pack, deg_day, melt, prec_effect, rain, snow)

Snow melting and accumulation.

15.86.1 Detailed Description

Snow melting and accumulation.

This module calculates snow melting and accumulation.

Authors

Vladyslav Prykhodko

Date

Dec 2012

15.86.2 Function/Subroutine Documentation

15.86.2.1 snow_accum_melt()

```
subroutine, public mo_snow_accum_melt::snow_accum_melt (
    real(dp), intent(in) deg_day_incr,
    real(dp), intent(in) deg_day_max,
    real(dp), intent(in) deg_day_noprec,
    real(dp), intent(in) prec,
    real(dp), intent(in) temperature,
    real(dp), intent(in) temperature_thresh,
    real(dp), intent(in) thrfall,
    real(dp), intent(inout) snow_pack,
    real(dp), intent(out) deg_day,
    real(dp), intent(out) melt,
    real(dp), intent(out) prec_effect,
    real(dp), intent(out) rain,
    real(dp), intent(out) snow )
```

Snow melting and accumulation.

Separates throughfall into rain and snow by comparing the temperature with the threshold. by comparing the temperature with the threshold. Calculates degree daily factor. Calculates snow melting rates. Calculates snow, rain and effective precipitation depth and snow pack.

Parameters

in	<i>REAL(dp) :: deg_day_incr</i>	Increase of the Degree-day factor per mm of increase in precipitation [s-1 degreeC-1]
in	<i>REAL(dp) :: deg_day_max</i>	Maximum Degree-day factor [m-1 degreeC-1]
in	<i>REAL(dp) :: deg_day_noprec</i>	Degree-day factor with no precipitation [m-1 degreeC-1]
in	<i>REAL(dp) :: prec</i>	Daily mean precipitation [m]
in	<i>REAL(dp) :: temperature</i>	Daily mean temperature [degreeC]
in	<i>REAL(dp) :: temperature_thresh</i>	Threshold temperature for snow/rain [degreeC]
in	<i>REAL(dp) :: thrfall</i>	Throughfall [m s-1]
in, out	<i>REAL(dp) :: snow_pack</i>	Snow pack [m]
out	<i>REAL(dp) :: deg_day</i>	Degree-day factor [m s-1 degreeC-1]
out	<i>REAL(dp) :: melt</i>	Melting snow depth [m s-1]
out	<i>REAL(dp) :: prec_effect</i>	Effective precipitation depth (snow melt + rain) [m]
out	<i>REAL(dp) :: rain</i>	Rain precipitation depth [m]
out	<i>REAL(dp) :: snow</i>	Snow precipitation depth [m]

Authors

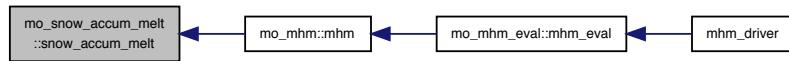
Vladyslav Prykhodko

Date

Dec 2012

Referenced by `mo_mhm::mhm()`.

Here is the caller graph for this function:



15.87 mo_soil_database Module Reference

Generating soil database from input file.

Functions/Subroutines

- subroutine, public `read_soil_lut` (filename)
Reads the soil LUT file.
- subroutine, public `generate_soil_database`
Generates soil database.

15.87.1 Detailed Description

Generating soil database from input file.

This module provides the routines for generating the soil database for mHM from an ASCII input file. One routine `read_soil_LUT` reads a soil LookUpTable, performs some consistency checks and returns an initial soil database. The second routine `generate_soil_database` calculates based on the initial one the proper soil database.

Authors

Juliane Mai

Date

Dec 2012

15.87.2 Function/Subroutine Documentation

15.87.2.1 generate_soil_database()

```
subroutine, public mo_soil_database::generate_soil_database ( )
```

Generates soil database.

Calculates the proper soil database using the initialized soil database from `read_soil_LUT`.

Authors

Juliane Mai

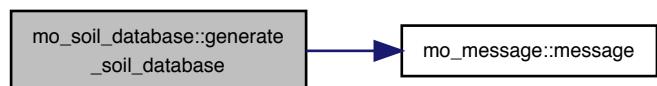
Date

Dec 2012

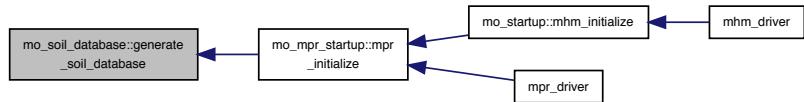
References `mo_mpr_global_variables::horizondepth_mhm`, `mo_mpr_global_variables::iflag_soildb`, `mo_message::message()`, `mo_common_constants::nodata_dp`, `mo_common_constants::nodata_i4`, `mo_mpr_global_variables::nsoilhorizons_mhm`, `mo_mpr_global_variables::nsoiltypes`, and `mo_mpr_global_variables::soildb`.

Referenced by `mo_mpr_startup::mpr_initialize()`.

Here is the call graph for this function:



Here is the caller graph for this function:



15.87.2.2 read_soil_lut()

```
subroutine, public mo_soil_database::read_soil_lut (
    character(len = *), intent(in) filename )
```

Reads the soil LUT file.

Reads the soil LookUpTable file and checks for consistency.

Parameters

in	character(len = *) :: filename	filename of the soil LUT
----	--------------------------------	--------------------------

Authors

Juliane Mai

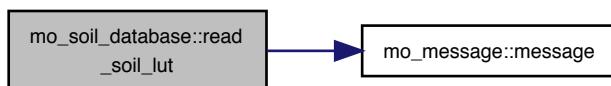
Date

Dec 2012

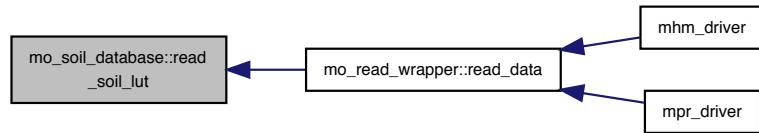
References mo_kind::dp, mo_common_constants::eps_dp, mo_mpr_global_variables::horizondepth_mhm, mo_mpr_global_variables::iflag_soildb, mo_message::message(), mo_mpr_constants::nlcover_class, mo_common_constants::nodata_dp, mo_common_constants::nodata_i4, mo_mpr_global_variables::nsoilhorizons_mhm, mo_mpr_global_variables::nsoiltypes, mo_mpr_global_variables::soildb, mo_mpr_global_variables::tillagedepth, and mo_mpr_file::usoil_database.

Referenced by mo_read_wrapper::read_data().

Here is the call graph for this function:



Here is the caller graph for this function:



15.88 mo_soil_moisture Module Reference

Soil moisture of the different layers.

Functions/Subroutines

- subroutine, public [soil_moisture](#) (processCase, frac_sealed, water_thresh_sealed, pet, evap_coeff, soil_moist_sat, frac_roots, soil_moist_FC, wilting_point, soil_moist_exponen, jarvis_thresh_c1, aet_canopy, prec_effec, runoff_sealed, storage_sealed, infiltration, soil_moist, aet, aet_sealed)
 - Soil moisture in different soil horizons.*
- elemental pure real(dp) function, private [feddes_et_reduction](#) (soil_moist, soil_moist_FC, wilting_point, frac_roots)
 - stress factor for reducing evapotranspiration based on actual soil moisture*
- elemental pure real(dp) function, private [jarvis_et_reduction](#) (soil_moist, soil_moist_sat, wilting_point, frac_roots, jarvis_thresh_c1)
 - stress factor for reducing evapotranspiration based on actual soil moisture*

15.88.1 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 - `processCase(1)`: $f_{SM} = \frac{\theta - \theta_{pwp}}{\theta_{fc} - \theta_{pwp}}$ or using the Jarvis equation - `processCase(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

15.88.2 Function/Subroutine Documentation

15.88.2.1 feddes_et_reduction()

```
elemental pure real(dp) function, private mo_soil_moisture::feddes_et_reduction (
    real(dp), intent(in) soil_moist,
    real(dp), intent(in) soil_moist_FC,
    real(dp), intent(in) wilting_point,
    real(dp), intent(in) frac_roots ) [private]
```

stress factor for reducing evapotranspiration based on actual soil moisture

Potential evapotranspiration is reduced to 0 if SM is lower PWP. PET is equal fraction of roots if soil moisture is exceeding field capacity. If soil moisture is in between PWP and FC PET is reduced by fraction of roots times a stress factor. The ET reduction factor f is estimated as

$$f = \begin{cases} f_{roots} & \text{if } \theta \geq \theta_{fc} \\ f_{roots} \cdot \frac{\theta - \theta_{pwp}}{\theta_{fc} - \theta_{pwp}} & \text{if } \theta < \theta_{fc} \\ 0 & \text{if } \theta < \theta_{pwp} \end{cases}$$

Parameters

in	real(dp) :: soil_moist	Soil moisture of each horizon [mm]
in	real(dp) :: soil_moist_FC	Soil moisture below which actual ET is reduced [mm]
in	real(dp) :: wilting_point	Permanent wilting point
in	real(dp) :: frac_roots	Fraction of Roots in soil horizon is reduced [mm]

Returns

real(dp) :: feddes_et_reduction; et reduction factor

Authors

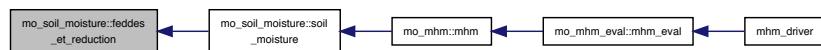
Matthias Cuntz, Cueneyd Demirel, Matthias Zink

Date

March 2017

Referenced by soil_moisture().

Here is the caller graph for this function:



15.88.2.2 jarvis_et_reduction()

```
elemental pure real(dp) function, private mo_soil_moisture::jarvis_et_reduction (
    real(dp), intent(in) soil_moist,
    real(dp), intent(in) soil_moist_sat,
```

```
real(dp), intent(in) wilting_point,
real(dp), intent(in) frac_roots,
real(dp), intent(in) jarvis_thresh_c1 ) [private]
```

stress factor for reducing evapotranspiration based on actual soil moisture

The soil moisture stress factor is estimated based on the normalized soil water content θ_{norm} is estimated as:

$$\theta_{norm} = \frac{\theta - \theta_{pwp}}{\theta_{sat} - \theta_{pwp}}$$

The ET reduction factor f is estimated as

$$f = \begin{cases} f_{roots} & \text{if } \theta_{norm} \geq \text{jarvis_sm_threshold_c1} \\ f_{roots} \frac{\theta_{norm}}{\text{jarvis_sm_threshold_c1}} & \text{if } \theta_{norm} < \text{jarvis_sm_threshold_c1} \end{cases}$$

Parameters

in	real(dp) :: soil_moist	Soil moisture of each horizon [mm]
in	real(dp) :: soil_moist_sat	saturated Soil moisture content [mm]
in	real(dp) :: wilting_point	Permanent wilting point
in	real(dp) :: frac_roots	Fraction of Roots in soil horizon is reduced [mm]
in	real(dp) :: jarvis_thresh_c1	parameter C1 from Jarvis formulation

Returns

real(dp) :: jarvis_et_reduction; et reduction factor

Authors

Cueneyd Demirel, Matthias Zink

Date

March 2017

Referenced by soil_moisture().

Here is the caller graph for this function:



15.88.2.3 soil_moisture()

```
subroutine, public mo_soil_moisture::soil_moisture (
    integer(i4), intent(in) processCase,
    real(dp), intent(in) frac_sealed,
    real(dp), intent(in) water_thresh_sealed,
    real(dp), intent(in) pet,
```

```

real(dp), intent(in) evap_coeff,
real(dp), dimension(:), intent(in) soil_moist_sat,
real(dp), dimension(:), intent(in) frac_roots,
real(dp), dimension(:), intent(in) soil_moist_FC,
real(dp), dimension(:), intent(in) wilting_point,
real(dp), dimension(:), intent(in) soil_moist_exponen,
real(dp), intent(in) jarvis_thresh_c1,
real(dp), intent(in) aet_canopy,
real(dp), intent(inout) prec_effec,
real(dp), intent(inout) runoff_sealed,
real(dp), intent(inout) storage_sealed,
real(dp), dimension(size(soil_moist_sat, 1)), intent(inout) infiltration,
real(dp), dimension(size(soil_moist_sat, 1)), intent(inout) soil_moist,
real(dp), dimension(size(soil_moist_sat, 1)), intent(out) aet,
real(dp), intent(out) aet_sealed

```

Soil moisture in different soil horizons.

Infiltration I from one layer $k - 1$ to the next k on pervious areas is calculated as (omit t)

$$I[k] = I[k - 1](\theta[k]/\theta_{sat}[k])^{\beta[k]}$$

Then soil moisture can be calculated as (omit k)

$$\theta[t] = \theta[t - 1] + I[t] - ET[t]$$

with ET (omit $[k, t]$) being

$$ET = f_{\text{roots}} \cdot f_{SM} \cdot PET$$

Parameters

in	integer(i4) :: processCase	1 - Feddes equation for PET reduction 2 - Jarvis equation for PET reduction 3 - 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	Threshold water depth in impervious areas [mm/s]
in	real(dp) :: pet	Reference evapotranspiration [mm/s]
in	real(dp) :: evap_coeff	Evaporation coefficient 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 or effective precipitation of each horizon [mm/s]

Parameters

in, out	<i>real(dp), dimension(size(soil_moist_sat, 1)) :: soil_moist</i>	Soil moisture of each horizon [mm]
out	<i>real(dp), dimension(size(soil_moist_sat, 1)) :: aet</i>	actual ET [mm/s]
out	<i>real(dp) :: aet_sealed</i>	actual ET from free-water surfaces,i.e impervious cover [mm/s]

Authors

Matthias Cuntz

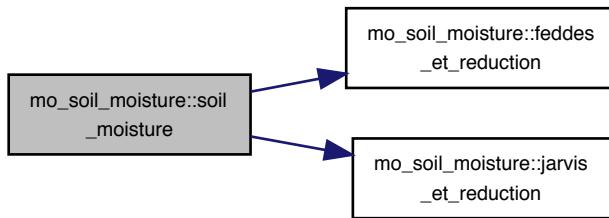
Date

Dec 2012

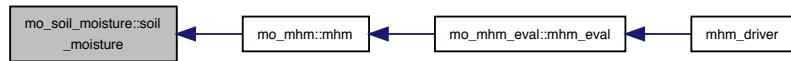
References `mo_common_constants::eps_dp`, `feddes_et_reduction()`, and `jarvis_et_reduction()`.

Referenced by `mo_mhm::mhm()`.

Here is the call graph for this function:



Here is the caller graph for this function:



15.89 mo_spatial_agg_disagg_forcing Module Reference

Spatial aggregation or disaggregation of meteorological input data.

Data Types

- interface [spatial_aggregation](#)

- Spatial aggregation of meterological variables.*
- interface [spatial_disaggregation](#)
- Spatial disaggregation of meterological variables.*

Functions/Subroutines

- subroutine [spatial_aggregation_3d](#) (data2, cellsize2, cellsize1, mask1, mask2, data1)
- subroutine [spatial_aggregation_4d](#) (data2, cellsize2, cellsize1, mask1, mask2, data1)
- subroutine [spatial_disaggregation_3d](#) (data2, cellsize2, cellsize1, mask1, mask2, data1)
- subroutine [spatial_disaggregation_4d](#) (data2, cellsize2, cellsize1, mask1, mask2, data1)

15.89.1 Detailed Description

Spatial aggregation or disaggregation of meteorological input data.

This module contains two subroutines to upscale and downscale, respectively, the level-2 meteorological inputs to a required Level-1 hydrological spatial resolution.

Authors

Rohini Kumar

Date

Jan 2013

15.89.2 Function/Subroutine Documentation

15.89.2.1 [spatial_aggregation_3d\(\)](#)

```
subroutine mo_spatial_agg_disagg_forcing::spatial_aggregation_3d (
    real(dp), dimension(:, :, :), intent(in) data2,
    real(dp), intent(in) cellsize2,
    real(dp), intent(in) cellsize1,
    logical, dimension(:, :, :), intent(in) mask1,
    logical, dimension(:, :, :), intent(in) mask2,
    real(dp), dimension(:, :, :, :), intent(out), allocatable data1 ) [private]
```

References [mo_common_constants::nodata_dp](#).

15.89.2.2 [spatial_aggregation_4d\(\)](#)

```
subroutine mo_spatial_agg_disagg_forcing::spatial_aggregation_4d (
    real(dp), dimension(:, :, :, :, :), intent(in) data2,
    real(dp), intent(in) cellsize2,
    real(dp), intent(in) cellsize1,
    logical, dimension(:, :, :), intent(in) mask1,
    logical, dimension(:, :, :), intent(in) mask2,
    real(dp), dimension(:, :, :, :, :), intent(out), allocatable data1 )
```

References [mo_common_constants::nodata_dp](#).

15.89.2.3 spatial_disaggregation_3d()

```
subroutine mo_spatial_agg_disagg_forcing::spatial_disaggregation_3d (
    real(dp), dimension(:, :, :), intent(in) data2,
    real(dp), intent(in) cellsize2,
    real(dp), intent(in) cellsize1,
    logical, dimension(:, :, :), intent(in) mask1,
    logical, dimension(:, :, :), intent(in) mask2,
    real(dp), dimension(:, :, :), intent(out), allocatable data1 )
```

References mo_common_constants::nodata_dp.

15.89.2.4 spatial_disaggregation_4d()

```
subroutine mo_spatial_agg_disagg_forcing::spatial_disaggregation_4d (
    real(dp), dimension(:, :, :, :), intent(in) data2,
    real(dp), intent(in) cellsize2,
    real(dp), intent(in) cellsize1,
    logical, dimension(:, :, :), intent(in) mask1,
    logical, dimension(:, :, :), intent(in) mask2,
    real(dp), dimension(:, :, :, :), intent(out), allocatable data1 )
```

References mo_common_constants::nodata_dp.

15.90 mo_spatialsimilarity Module Reference

Routines for bias insensitive comparison of spatial patterns.

Data Types

- interface [nndv](#)
Calculates the number of neighboring dominating values, a measure for spatial dissimilarity.
- interface [pd](#)
Calculates pattern dissimilarity (PD) measure.

Functions/Subroutines

- real(sp) function [nndv_sp](#) (mat1, mat2, mask, valid)
- real(dp) function [nndv_dp](#) (mat1, mat2, mask, valid)
- real(sp) function [pd_sp](#) (mat1, mat2, mask, valid)
- real(dp) function [pd_dp](#) (mat1, mat2, mask, valid)

15.90.1 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

15.90.2 Function/Subroutine Documentation

15.90.2.1 nndv_dp()

```
real(dp) function mo_spatialsimilarity::nndv_dp (
    real(dp), dimension(:, :, ), intent(in) mat1,
    real(dp), dimension(:, :, ), intent(in) mat2,
    logical, dimension(:, :, ), intent(in), optional mask,
    logical, intent(out), optional valid )
```

15.90.2.2 nndv_sp()

```
real(sp) function mo_spatialsimilarity::nndv_sp (
    real(sp), dimension(:, :, ), intent(in) mat1,
    real(sp), dimension(:, :, ), intent(in) mat2,
    logical, dimension(:, :, ), intent(in), optional mask,
    logical, intent(out), optional valid )
```

15.90.2.3 pd_dp()

```
real(dp) function mo_spatialsimilarity::pd_dp (
    real(dp), dimension(:, :, ), intent(in) mat1,
    real(dp), dimension(:, :, ), intent(in) mat2,
    logical, dimension(:, :, ), intent(in), optional mask,
    logical, intent(out), optional valid )
```

References mo_kind::dp.

15.90.2.4 pd_sp()

```
real(sp) function mo_spatialsimilarity::pd_sp (
    real(sp), dimension(:, :, ), intent(in) mat1,
    real(sp), dimension(:, :, ), intent(in) mat2,
    logical, dimension(:, :, ), intent(in), optional mask,
    logical, intent(out), optional valid )
```

References mo_kind::sp.

15.91 mo_standard_score Module Reference

Routines for calculating the normalization (anomaly)/standard score/z score and the deseasonalized (standard score on monthly basis) values of a time series.

Data Types

- interface [classified_standard_score](#)
Calculates the classified standard score (e.g. classes are months).
- interface [standard_score](#)
Calculates the standard score / normalization (anomaly) / z-score.

Functions/Subroutines

- real(sp) function, dimension(size(data, dim=1)) [standard_score_sp](#) (data, mask)
- real(dp) function, dimension(size(data, dim=1)) [standard_score_dp](#) (data, mask)
- real(sp) function, dimension(size(data, dim=1)) [classified_standard_score_sp](#) (data, classes, mask)
- real(dp) function, dimension(size(data, dim=1)) [classified_standard_score_dp](#) (data, classes, mask)

15.91.1 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

15.91.2 Function/Subroutine Documentation

15.91.2.1 [classified_standard_score_dp\(\)](#)

```
real(dp) function, dimension(size(data, dim = 1)) mo_standard_score::classified_standard_←
score_dp (
    real(dp), dimension(:), intent(in) data,
    integer, dimension(:), intent(in) classes,
    logical, dimension(:), intent(in), optional mask )
```

15.91.2.2 [classified_standard_score_sp\(\)](#)

```
real(sp) function, dimension(size(data, dim = 1)) mo_standard_score::classified_standard_←
score_sp (
    real(sp), dimension(:), intent(in) data,
    integer, dimension(:), intent(in) classes,
    logical, dimension(:), intent(in), optional mask )
```

15.91.2.3 standard_score_dp()

```
real(dp) function, dimension(size(data, dim = 1)) mo_standard_score::standard_score_dp (
    real(dp), dimension(:), intent(in) data,
    logical, dimension(:), intent(in), optional mask )
```

15.91.2.4 standard_score_sp()

```
real(sp) function, dimension(size(data, dim = 1)) mo_standard_score::standard_score_sp (
    real(sp), dimension(:), intent(in) data,
    logical, dimension(:), intent(in), optional mask ) [private]
```

15.92 mo_startup Module Reference

Startup procedures for mHM.

Functions/Subroutines

- subroutine, public [mhm_initialize](#)
Initialize main mHM variables.
- subroutine [constants_init](#)
Initialize mHM constants.
- subroutine [l2_variable_init](#) (iBasin, level0_iBasin, level2_iBasin)
Initialize Level-2 meteorological forcings data.

15.92.1 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

15.92.2 Function/Subroutine Documentation

15.92.2.1 constants_init()

```
subroutine mo_startup::constants_init ( )
```

Initialize mHM constants.

transformation of time units & initialize constants

Authors

Luis Samaniego

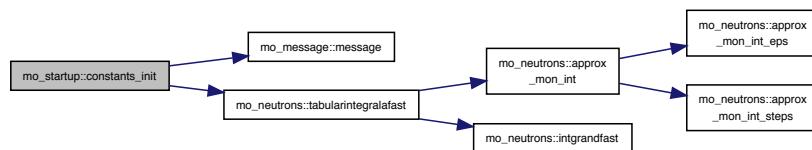
Date

Dec 2012

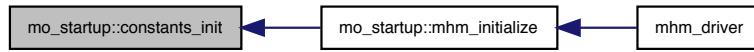
References `mo_mpr_global_variables::c2stu`, `mo_mpr_file::file_hydrogeoclass`, `mo_file::file_namelist_mhm::param`, `mo_mpr_global_variables::geounitlist`, `mo_message::message()`, `mo_global_variables::neutron_integral::afast`, `mo_common_variables::processmatrix`, `mo_neutrons::tabularintegralafast()`, and `mo_common_mhm_mrm::variables::timestep`.

Referenced by `mhm_initialize()`.

Here is the call graph for this function:



Here is the caller graph for this function:



15.92.2.2 l2_variable_init()

```

subroutine mo_startup::l2_variable_init (
    integer(i4), intent(in)  iBasin,
    type(grid), intent(in)  level0_iBasin,
    type(grid), intent(inout) level2_iBasin )
  
```

Initialize Level-2 meteorological forcings data.

following tasks are performed 1) cell id & numbering 2) mask creation 3) append variable of interest to global ones

Parameters

in	<code>integer(i4) :: iBasin</code>	Basin Id
in	<code>type(Grid) :: level0_iBasin</code>	
in,out	<code>type(Grid) :: level2_iBasin</code>	

Authors

Rohini Kumar

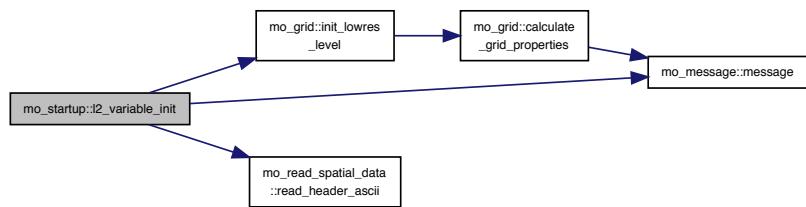
Date

Feb 2013

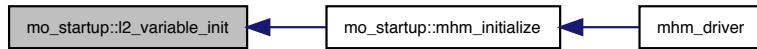
References `mo_global_variables::dirprecipitation`, `mo_mpr_file::file_meteo_header`, `mo_grid::init_lowres_level()`, `mo_message::message()`, `mo_read_spatial_data::read_header_ascii()`, and `mo_mpr_file::umeteo_header`.

Referenced by `mhm_initialize()`.

Here is the call graph for this function:



Here is the caller graph for this function:



15.92.2.3 mhm_initialize()

subroutine, public `mo_startup::mhm_initialize ()`

Initialize main mHM variables.

Initialize main mHM variables for a given basin. Calls the following procedures in this order:

- Constant initialization.
- Generate soil database.
- Checking inconsistencies input fields.
- Variable initialization at level-0.
- Variable initialization at level-1.
- Variable initialization at level-11.
- Space allocation of remaining variable/parameters. Global variables will be used at this stage.

Authors

Luis Samaniego, Rohini Kumar

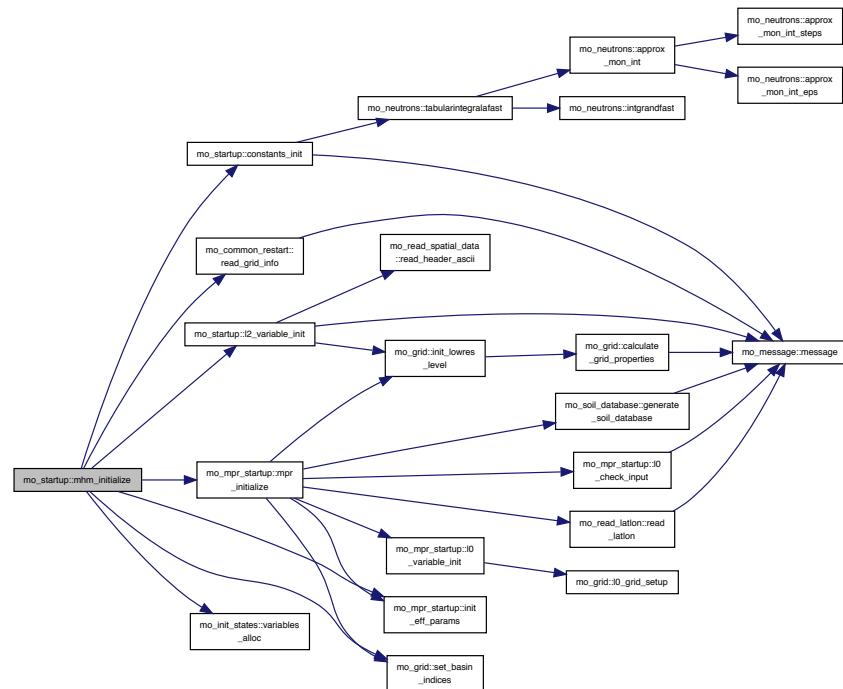
Date

Dec 2012

References constants_init(), mo_common_mhm_mrm_variables::dirrestartin, mo_kind::i4, mo_mpr_startup::init_eff_params(), mo_common_variables::l0_basin, l2_variable_init(), mo_common_variables::level0, mo_common_variables::level1, mo_global_variables::level2, mo_mpr_startup::mpr_initialize(), mo_common_variables::nbasins, mo_common_restart::read_grid_info(), mo_common_mhm_mrm_variables::read_restart, mo_grid::set_basin_indices(), and mo_init_states::variables_alloc().

Referenced by mhm_driver().

Here is the call graph for this function:



Here is the caller graph for this function:



15.93 mo_string_utils Module Reference

String utilities.

Data Types

- interface `num2str`
Convert to string.
- interface `numarray2str`
Convert to string.

Functions/Subroutines

- character(len(whitespaces)) function, public `compress` (whiteSpaces, n)
- subroutine, public `divide_string` (string, delim, strArr)
Divide string in substrings.
- logical function, public `equalstrings` (string1, string2)
- logical function, public `nonnull` (str)
Checks if string was already used.
- character(len=256) function, dimension(:), allocatable, public `splitstring` (string, delim)
- logical function, public `startswith` (string, start)
- character(len=len_trim(upper)) function, public `tolower` (upper)
Convert to lower case.
- character(len=len_trim(lower)) function, public `toupper` (lower)
- pure character(len=10) function `i42str` (nn, form)
- pure character(len=20) function `i82str` (nn, form)
- pure character(len=32) function `sp2str` (rr, form)
- pure character(len=32) function `dp2str` (rr, form)
- pure character(len=10) function `log2str` (ll, form)
- character(len=size(arr)) function `i4array2str` (arr)
- integer(i4) function, dimension(:), allocatable, public `str2num` (string)

Variables

- character(len=*), parameter, public `separator` = repeat('-', 70)

15.93.1 Detailed Description

String utilities.

This module provides string conversion and checking utilities.

Authors

Matthias Cuntz, Matthias Zink, Giovanni Dalmasso, David Schaefer

Date

Dec 2011

15.93.2 Function/Subroutine Documentation

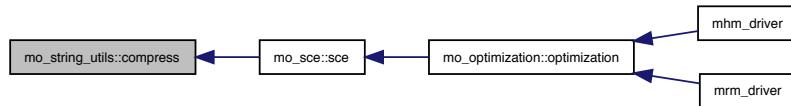
15.93.2.1 compress()

```
character(len(whitespaces)) function, public mo_string_utils::compress (
    character(len = *), intent(in) whiteSpaces,
    integer(i4), intent(out), optional n )
```

References mo_kind::i4.

Referenced by mo_sce::sce().

Here is the caller graph for this function:



15.93.2.2 divide_string()

```
subroutine, public mo_string_utils::divide_string (
    character(len = *), intent(in) string,
    character(len = *), intent(in) delim,
    character(len = *), dimension(:), intent(out), allocatable strArr )
```

Divide string in substrings.

Divides a string in several substrings (array of strings) with the help of a user specified delimiter.

Parameters

in	CHARACTER(len=*), INTENT(IN) :: string	- string to be divided
in	CHARACTER(len=*), INTENT(IN) :: delim	- delimiter specifying places for division
out	CHARACTER(len=*), DIMENSION(:), ALLOCATABLE, INTENT(OUT) :: strArr	Array of substrings, has to be allocatable and is handed to the routine unallocated

Author

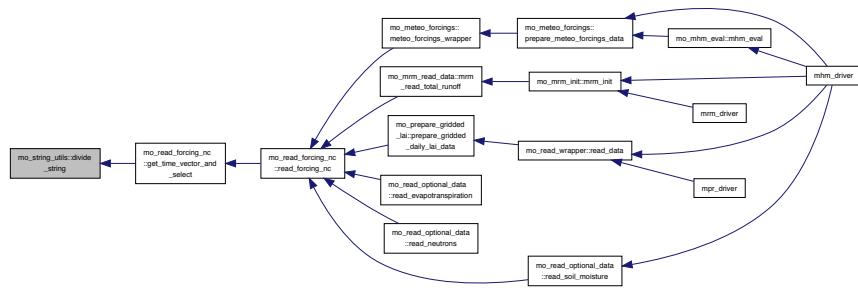
Matthias Zink

Date

Oct 2012

Referenced by `mo_read_forcing_nc::get_time_vector_and_select()`.

Here is the caller graph for this function:



15.93.2.3 dp2str()

```
pure character(len = 32) function mo_string_utils::dp2str (
    real(dp), intent(in) rr,
    character(len = *), intent(in), optional form ) [private]
```

15.93.2.4 equalstrings()

```
logical function, public mo_string_utils::equalstrings (
```

character(len = *), intent(in) string1,	character(len = *), intent(in) string2)
---	--

References str2num().

Here is the call graph for this function:



15.93.2.5 i42str()

```
pure character(len = 10) function mo_string_utils::i42str (  
    integer(i4), intent(in) nn,  
    character(len = *), intent(in), optional form ) [private]
```

15.93.2.6 i4array2str()

```
character(len = size(arr)) function mo_string_utils::i4array2str (
    integer(i4), dimension(:), intent(in) arr ) [private]
```

15.93.2.7 i82str()

```
pure character(len = 20) function mo_string_utils::i82str (
    integer(i8), intent(in) nn,
    character(len = *), intent(in), optional form ) [private]
```

15.93.2.8 log2str()

```
pure character(len = 10) function mo_string_utils::log2str (
    logical, intent(in) ll,
    character(len = *), intent(in), optional form ) [private]
```

15.93.2.9 nonull()

```
logical function, public mo_string_utils::nonull (
    character(len = *), intent(in) str )
```

Checks if string was already used.

Checks if string was already used, i.e. does not contain NULL character anymore.

Parameters

in	character(len=*) :: str	String
----	-------------------------	--------

Returns

logical :: used — .true.: string was already set; .false.: string still in initialised state

Author

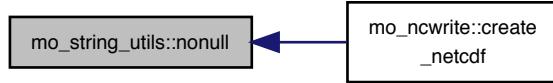
Matthias Cuntz

Date

Jan 2012

Referenced by `mo_ncwrite::create_netcdf()`.

Here is the caller graph for this function:



15.93.2.10 sp2str()

```

pure character(len = 32) function mo_string_utils::sp2str (
    real(sp), intent(in) rr,
    character(len = *), intent(in), optional form ) [private]

```

15.93.2.11 splitstring()

```

character(len = 256) function, dimension(:), allocatable, public mo_string_utils::splitstring
(
    character(len = *), intent(in) string,
    character(len = *), intent(in) delim )

```

References `str2num()`.

Here is the call graph for this function:



15.93.2.12 startswith()

```

logical function, public mo_string_utils::startswith (
    character(len = *), intent(in) string,
    character(len = *), intent(in) start )

```

References str2num().

Here is the call graph for this function:

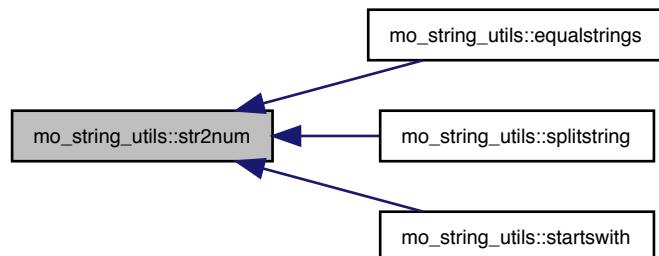


15.93.2.13 str2num()

```
integer(i4) function, dimension(:), allocatable, public mo_string_utils::str2num ( character(len = *), intent(in) string )
```

Referenced by equalstrings(), splitstring(), and startswith().

Here is the caller graph for this function:



15.93.2.14 tolower()

```
character(len = len_trim(upper)) function, public mo_string_utils::tolower ( character(len = *), intent(in) upper )
```

Convert to lower case.

Convert all upper case letters in string to lower case letters.

Parameters

in	character(len=*) :: upper	String
----	---------------------------	--------

Returns

```
character(len=len_trim(upper)) :: low — String where all uppercase in input is converted to lowercase
```

Author

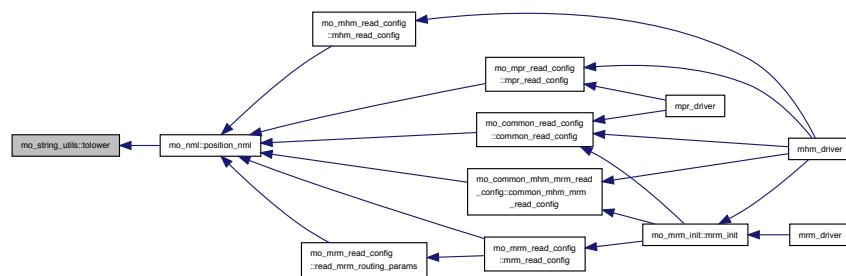
Matthias Cuntz - modified from Echam5, (C) MPI-MET, Hamburg, Germany

Date

Dec 2011

Referenced by `mo_nml::position_nml()`.

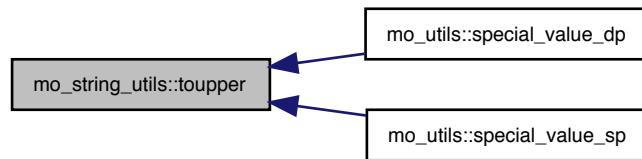
Here is the caller graph for this function:

**15.93.2.15 toupper()**

```
character(len = len_trim(lower)) function, public mo_string_utils::toupper (
    character(len = *), intent(in) lower )
```

Referenced by `mo_utils::special_value_dp()`, and `mo_utils::special_value_sp()`.

Here is the caller graph for this function:

**15.93.3 Variable Documentation**

15.93.3.1 separator

```
character(len = *), parameter, public mo_string_utils::separator = repeat('-', 70)
```

Referenced by `mo_finish::finish()`, `mhm_driver()`, and `mo_mrm_init::print_startup_message()`.

15.94 mo_template Module Reference

Template for future module developments.

Data Types

- interface `mean`

The average.

Functions/Subroutines

- elemental pure real(dp) function, public `circum` (radius)
Circumference of a circle.
- real(dp) function `mean_dp` (dat, mask)
- real(sp) function `mean_sp` (dat, mask)

Variables

- real(dp), parameter, public `pi_dp` = 3.141592653589793238462643383279502884197_dp
Constant Pi in double precision.
- real(sp), parameter, public `pi_sp` = 3.141592653589793238462643383279502884197_sp
Constant Pi in single precision.
- integer(i4), parameter `itest` = 1

15.94.1 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

15.94.2 Function/Subroutine Documentation

15.94.2.1 circum()

```
elemental pure real(dp) function, public mo_template::circum (
    real(dp), intent(in) radius )
```

Circumference of a circle.

Calculates the circumference of a circle

$$c = 2\pi r$$

Parameters

in	real(dp) :: radius	Radius
----	--------------------	--------

Returns

real(dp) :: circum — circumference of circle.

Authors

Matthias Cuntz

Date

Dec 2012

References pi_dp.

15.94.2.2 mean_dp()

```
real(dp) function mo_template::mean_dp (
    real(dp), dimension(:), intent(in) dat,
    logical, dimension(:), intent(in), optional mask ) [private]
```

15.94.2.3 mean_sp()

```
real(sp) function mo_template::mean_sp (
    real(sp), dimension(:), intent(in) dat,
    logical, dimension(:), intent(in), optional mask ) [private]
```

15.94.3 Variable Documentation

15.94.3.1 itest

```
integer(i4), parameter mo_template::itest = 1 [private]
```

15.94.3.2 pi_dp

```
real(dp), parameter, public mo_template::pi_dp = 3.141592653589793238462643383279502884197_dp
```

Constant Pi in double precision.

Referenced by circum().

15.94.3.3 pi_sp

```
real(sp), parameter, public mo_template::pi_sp = 3.141592653589793238462643383279502884197_sp
```

Constant Pi in single precision.

15.95 mo_temporal_aggregation Module Reference

Temporal aggregation for time series (averaging)

Data Types

- interface [day2mon_average](#)
Day-to-month average ([day2mon_average](#))
- interface [hour2day_average](#)
Hour-to-day average ([hour2day_average](#))

Functions/Subroutines

- subroutine [day2mon_average_dp](#) (daily_data, yearS, monthS, dayS, mon_avg, misval, rm_misval)
- subroutine [hour2day_average_dp](#) (hourly_data, yearS, monthS, dayS, hourS, day_avg, misval, rm_misval)

15.95.1 Detailed Description

Temporal aggregation for time series (averaging)

This module does temporal aggregation (averaging) of time series

Authors

Odrich Rakovec, Rohini Kumar

Date

October 2015

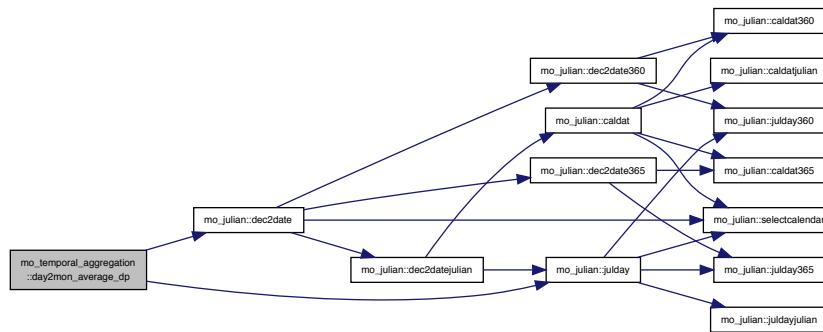
15.95.2 Function/Subroutine Documentation

15.95.2.1 day2mon_average_dp()

```
subroutine mo_temporal_aggregation::day2mon_average_dp (
    real(dp), dimension(:), intent(in) daily_data,
    integer(i4), intent(in) years,
    integer(i4), intent(in) months,
    integer(i4), intent(in) days,
    real(dp), dimension(:), intent(inout), allocatable mon_avg,
    real(dp), intent(in), optional misval,
    logical, intent(in), optional rm_misval ) [private]
```

References mo_julian::dec2date(), mo_common_constants::eps_dp, and mo_julian::julday().

Here is the call graph for this function:



15.95.2.2 hour2day_average_dp()

```
subroutine mo_temporal_aggregation::hour2day_average_dp (
    real(dp), dimension(:), intent(in) hourly_data,
    integer(i4), intent(in) years,
    integer(i4), intent(in) months,
    integer(i4), intent(in) days,
    integer(i4), intent(in) hours,
    real(dp), dimension(:), intent(inout), allocatable day_avg,
    real(dp), intent(in), optional misval,
    logical, intent(in), optional rm_misval ) [private]
```

15.96 mo_temporal_disagg_forcing Module Reference

Temporal disaggregation of daily input values.

Functions/Subroutines

- elemental pure subroutine, public [temporal_disagg_forcing](#) (isday, ntimesteps_day, prec_day, pet_day, temp←_day, fday_prec, fday_pet, fday_temp, fnight_prec, fnight_pet, fnight_temp, temp_weights, pet_weights, pre←_weights, read_meteo_weights, prec, pet, temp)

Temporally distribute daily mean forcings onto time step.

15.96.1 Detailed Description

Temporal disaggregation of daily input values.

Calculate actual values for precipitation, PET and temperature from daily mean inputs. There is not PET correction for aspect in this routine. Use $\text{pet} * \text{fasp}$ before or after the routine.

Authors

Matthias Cuntz

Date

Dec 2012

15.96.2 Function/Subroutine Documentation

15.96.2.1 temporal_disagg_forcing()

```
elemental pure subroutine, public mo_temporal_disagg_forcing::temporal_disagg_forcing (
    logical, intent(in) isday,
    real(dp), intent(in) ntimesteps_day,
    real(dp), intent(in) prec_day,
    real(dp), intent(in) pet_day,
    real(dp), intent(in) temp_day,
    real(dp), intent(in) fday_prec,
    real(dp), intent(in) fday_pet,
    real(dp), intent(in) fday_temp,
    real(dp), intent(in) fnight_prec,
    real(dp), intent(in) fnight_pet,
    real(dp), intent(in) fnight_temp,
    real(dp), intent(in) temp_weights,
    real(dp), intent(in) pet_weights,
    real(dp), intent(in) pre_weights,
    logical, intent(in) read_meteo_weights,
    real(dp), intent(out) prec,
    real(dp), intent(out) pet,
    real(dp), intent(out) temp )
```

Temporally distribute daily mean forcings onto time step.

Calculates actual precipitation, PET and temperature from daily mean inputs. Precipitation and PET are distributed with predefined factors onto the day. Temperature gets a predefined amplitude added on day and subtracted 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	<i>logical :: isday</i>	is day or night
in	<i>real(dp) :: ntimesteps_day</i>	# of time steps per day
in	<i>real(dp) :: prec_day</i>	Daily mean precipitation [mm/s]
in	<i>real(dp) :: pet_day</i>	Daily mean ET [mm/s]
in	<i>real(dp) :: temp_day</i>	Daily mean air temperature [K]
in	<i>real(dp) :: fday_prec</i>	Daytime fraction of precipitation
in	<i>real(dp) :: fday_pet</i>	Daytime fraction of PET

Parameters

in	<i>real(dp) :: fday_temp</i>	Daytime air temparture increase
in	<i>real(dp) :: fnight_prec</i>	Daytime fraction of precipitation
in	<i>real(dp) :: fnight_pet</i>	Daytime fraction of PET
in	<i>real(dp) :: fnight_temp</i>	Daytime air temparture increase
in	<i>real(dp) :: temp_weights</i>	weights for average temperature
in	<i>real(dp) :: pet_weights</i>	weights for PET
in	<i>real(dp) :: pre_weights</i>	weights for precipitation
in	<i>logical :: read_meteo_weights</i>	flag indicating that weights should be used
out	<i>real(dp) :: prec</i>	Actual precipitation [mm/s]
out	<i>real(dp) :: pet</i>	Reference ET [mm/s]
out	<i>real(dp) :: temp</i>	Air temperature [K]

Authors

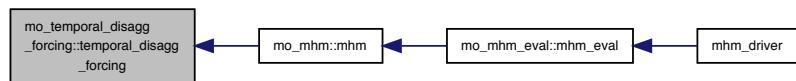
Matthias Cuntz

Date

Dec 2012

Referenced by `mo_mhm::mhm()`.

Here is the caller graph for this function:



15.97 mo_timer Module Reference

Timing routines.

Functions/Subroutines

- subroutine, public `timer_check` (timer)
Check a timer.
- subroutine, public `timer_clear` (timer)
Reset a timer.
- real(sp) function, public `timer_get` (timer)
Return a timer.
- subroutine, public `timer_print` (timer)
Print a timer.
- subroutine, public `timer_start` (timer)
Start a timer.
- subroutine, public `timer_stop` (timer)

- subroutine, public **timers_init**
Initialise timer module.

Variables

- integer(i4), parameter, public **max_timers** = 99
max number of timers allowed
- integer(i4), save, public **cycles_max**
max value of clock allowed by system
- real(sp), save, public **clock_rate**
clock_rate in seconds for each cycle
- integer(i4), dimension(**max_timers**), save, public **cycles1**
cycle number at start for each timer
- integer(i4), dimension(**max_timers**), save, public **cycles2**
cycle number at stop for each timer
- real(sp), dimension(**max_timers**), save, public **cputime**
accumulated cpu time in each timer
- character(len=8), dimension(**max_timers**), save, public **status**
timer status string

15.97.1 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 California

Date

Dec 2012

15.97.2 Function/Subroutine Documentation

15.97.2.1 timer_check()

```
subroutine, public mo_timer::timer_check (
    integer(i4), intent(in) timer )
```

Check a timer.

This routine checks a given timer. This is primarily used to periodically accumulate time in the timer to prevent timer cycles from wrapping around max_cycles.

Parameters

in	integer(i4) :: timer	timer number
----	----------------------	--------------

Author

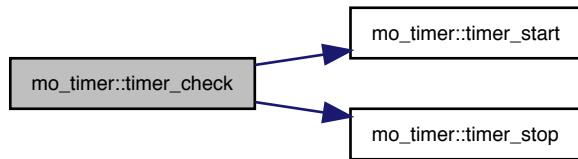
Matthias Cuntz

Date

Aug 2012

References `status`, `timer_start()`, and `timer_stop()`.

Here is the call graph for this function:



15.97.2.2 `timer_clear()`

```
subroutine, public mo_timer::timer_clear (
    integer(i4), intent(in), optional timer )
```

Reset a timer.

This routine resets a given timer or all timers to 0.

Parameters

in	<i>integer(i4), optional :: timer</i>	timer number if given. If missing, all timers will be reset.
----	---------------------------------------	---

Author

Matthias Cuntz

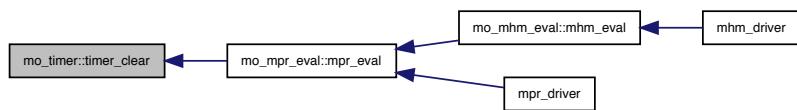
Date

Aug 2012

References cputime.

Referenced by mo_mpr_eval::mpr_eval().

Here is the caller graph for this function:



15.97.2.3 timer_get()

```
real(sp) function, public mo_timer::timer_get (
    integer(i4), intent(in) timer )
```

Return a timer.

This routine returns the result of a given timer. This can be called instead of timer_print so that the calling routine can print it in desired format.

Parameters

in	<i>integer(i4) :: timer</i>	timer number
----	-----------------------------	--------------

Author

Matthias Cuntz

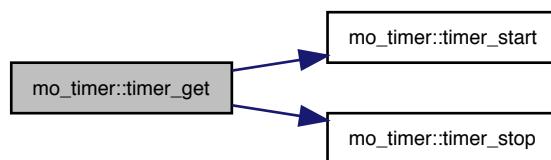
Date

Aug 2012

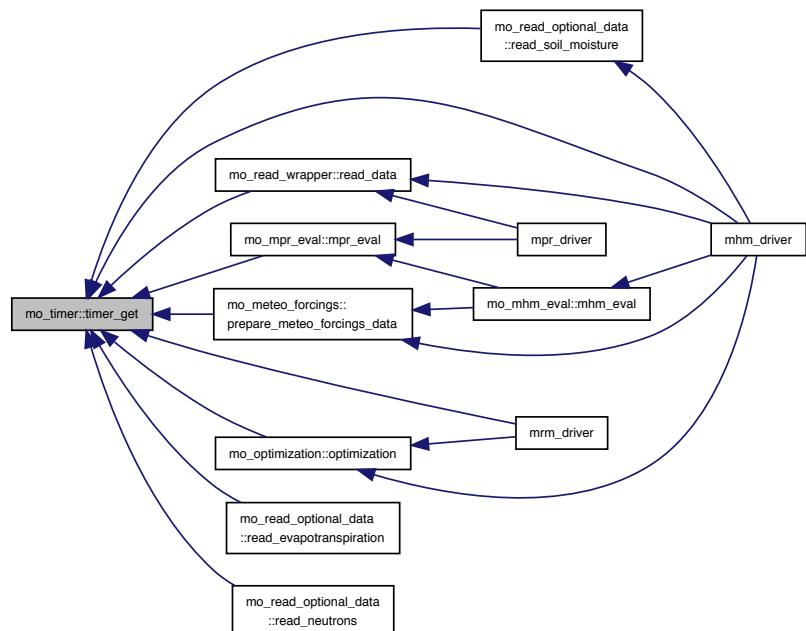
References cputime, status, timer_start(), and timer_stop().

Referenced by mhm_driver(), mo_mpr_eval::mpr_eval(), mrm_driver(), mo_optimization::optimization(), mo_meteo_forcings::prepare_meteo_forcings_data(), mo_read_wrapper::read_data(), mo_read_optional_data::read_evapotranspiration(), mo_read_optional_data::read_neutrons(), and mo_read_optional_data::read_soil_moisture().

Here is the call graph for this function:



Here is the caller graph for this function:



15.97.2.4 timer_print()

```
subroutine, public mo_timer::timer_print (
```

```
integer(i4), intent(in) timer )
```

Print a timer.

This routine prints the accumulated cpu time in given timer.

Parameters

in	integer(i4) :: timer	timer number
----	----------------------	--------------

Author

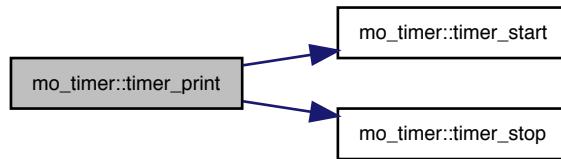
Matthias Cuntz

Date

Aug 2012

References cputime, status, timer_start(), and timer_stop().

Here is the call graph for this function:



15.97.2.5 timer_start()

```
subroutine, public mo_timer::timer_start (
    integer(i4), intent(in) timer )
```

Start a timer.

This routine starts a given timer.

Parameters

in	integer(i4) :: timer	timer number
----	----------------------	--------------

Author

Matthias Cuntz

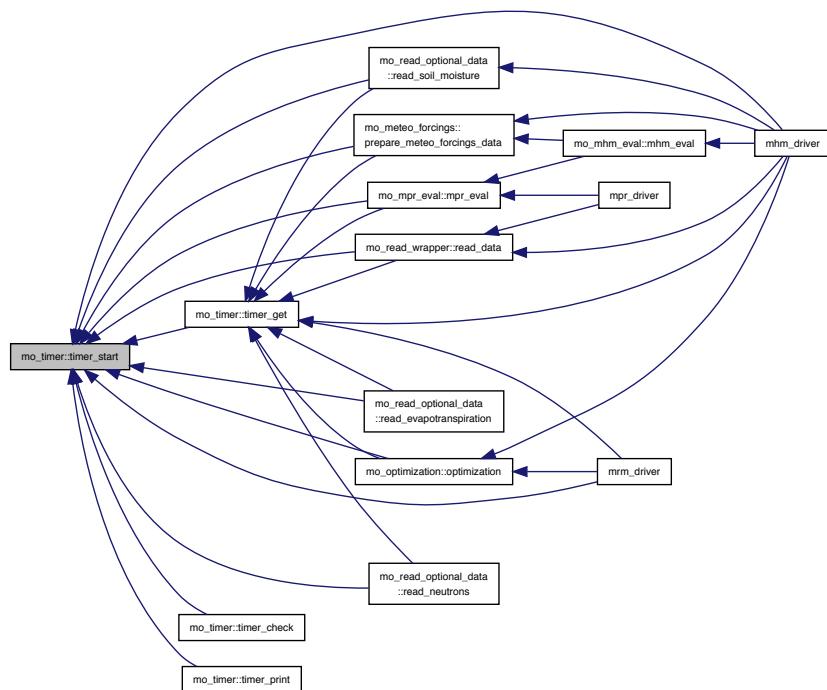
Date

Aug 2012

References cycles1, and status.

Referenced by `mhmm_driver()`, `mo_mpr_eval::mpr_eval()`, `mrm_driver()`, `mo_optimization::optimization()`, `mo_meteo_forcings::prepare_meteo_forcings_data()`, `mo_read_wrapper::read_data()`, `mo_read_optional_data::read_evapotranspiration()`, `mo_read_optional_data::read_neutrons()`, `mo_read_optional_data::read_soil_moisture()`, `timer_check()`, `timer_get()`, and `timer_print()`.

Here is the caller graph for this function:



15.97.2.6 timer_stop()

```
subroutine, public mo_timer::timer_stop (
    integer(i4), intent(in) timer )
```

Stop a timer.

This routine stops a given timer.

Parameters

in	<code>integer(i4) :: timer</code>	timer number
----	-----------------------------------	--------------

Author

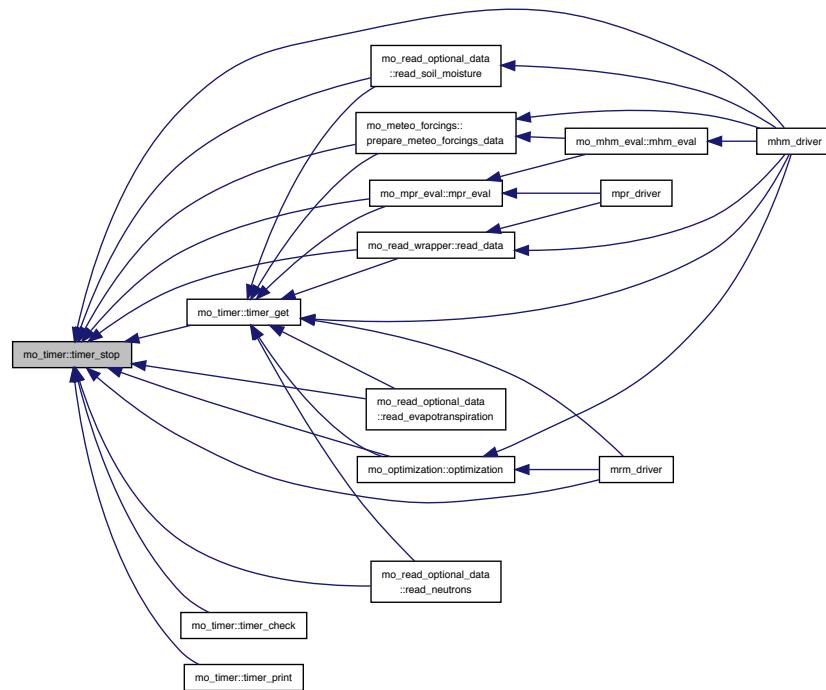
Matthias Cuntz

Date

Aug 2012

References `clock_rate`, `cpu_time`, `cycles1`, `cycles2`, `cycles_max`, and `status`.Referenced by `mhmm_driver()`, `mo_mpr_eval::mpr_eval()`, `mrm_driver()`, `mo_optimization::optimization()`, `mo_meteo_forcings::prepare_meteo_forcings_data()`, `mo_read_wrapper::read_data()`, `mo_read_optional_data::read_evapotranspiration()`, `mo_read_optional_data::read_neutrons()`, `mo_read_optional_data::read_soil_moisture()`, `timer_check()`, `timer_get()`, and `timer_print()`.

Here is the caller graph for this function:

15.97.2.7 `timers_init()`

```
subroutine, public mo_timer::timers_init ( )
```

Initialise timer module.

This routine initializes some machine parameters necessary for computing cpu time from F90 intrinsics.

Author

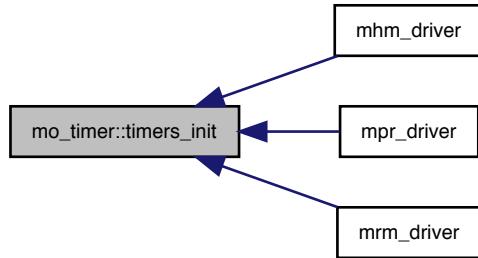
Matthias Cuntz

Date

Aug 2012

References `clock_rate`, `cpu_time`, `cycles1`, `cycles2`, `cycles_max`, and `status`.Referenced by `mhmm_driver()`, `mpr_driver()`, and `mrm_driver()`.

Here is the caller graph for this function:



15.97.3 Variable Documentation

15.97.3.1 clock_rate

`real(sp), save, public mo_timer::clock_rate`

`clock_rate` in seconds for each cycle

Referenced by `timer_stop()`, and `timers_init()`.

15.97.3.2 cputime

`real(sp), dimension(max_timers), save, public mo_timer::cputime`

accumulated cpu time in each timer

Referenced by `timer_clear()`, `timer_get()`, `timer_print()`, `timer_stop()`, and `timers_init()`.

15.97.3.3 cycles1

`integer(i4), dimension(max_timers), save, public mo_timer::cycles1`

cycle number at start for each timer

Referenced by `timer_start()`, `timer_stop()`, and `timers_init()`.

15.97.3.4 cycles2

`integer(i4), dimension(max_timers), save, public mo_timer::cycles2`

cycle number at stop for each timer

Referenced by `timer_stop()`, and `timers_init()`.

15.97.3.5 cycles_max

```
integer(i4), save, public mo_timer::cycles_max
```

max value of clock allowed by system

Referenced by timer_stop(), and timers_init().

15.97.3.6 max_timers

```
integer(i4), parameter, public mo_timer::max_timers = 99
```

max number of timers allowed

15.97.3.7 status

```
character(len = 8), dimension(max_timers), save, public mo_timer::status
```

timer status string

Referenced by timer_check(), timer_get(), timer_print(), timer_start(), timer_stop(), and timers_init().

15.98 mo_upscaling_operators Module Reference

Module containing upscaling operators.

Functions/Subroutines

- integer(i4) function, dimension(size(l1_upper_rowid_cell, 1)), public **majority_statistics** (nClass, L1_upper_rowid_cell, L1_lower_rowid_cell, L1_left_colonId_cell, L1_right_colonId_cell, L0_fineScale_2D_data)
majority statistics
- real(dp) function, dimension(size(l0upbound_inlx, 1)), public **l0_fractionalcover_in_lx** (dataIn0, classId, mask0, L0upBound_inLx, L0downBound_inLx, L0leftBound_inLx, L0rightBound_inLx, nTCells0_inLx)
fractional coverage of a given class of L0 fields in Lx field (Lx = L1 or L11)
- real(dp) function, dimension(size(nl0_cells_in_l1_cell, 1)), public **upscale_arithmetic_mean** (nL0_cells_in_L1_cell, L1_upper_rowid_cell, L1_lower_rowid_cell, L1_left_colonId_cell, L1_right_colonId_cell, L0_cellId, mask0, nodata_value, L0_fineScale_data)
arithmetic mean
- real(dp) function, dimension(size(nl0_cells_in_l1_cell, 1)), public **upscale_harmonic_mean** (nL0_cells_in_L1_cell, L1_upper_rowid_cell, L1_lower_rowid_cell, L1_left_colonId_cell, L1_right_colonId_cell, L0_cellId, mask0, nodata_value, L0_fineScale_data)
harmonic mean
- real(dp) function, dimension(size(l1_upper_rowid_cell, 1)), public **upscale_geometric_mean** (L1_upper_rowid_cell, L1_lower_rowid_cell, L1_left_colonId_cell, L1_right_colonId_cell, mask0, nodata_value, L0_fineScale_data)
geometric mean
- real(dp) function, dimension(size(nl0_cells_in_l1_cell, 1)) **upscale_p_norm** (nL0_cells_in_L1_cell, L1_upper_rowid_cell, L1_lower_rowid_cell, L1_left_colonId_cell, L1_right_colonId_cell, L0_cellId, mask0, nodata_value, p_norm, L0_fineScale_data)
arithmetic mean

15.98.1 Detailed Description

Module containing upscaling operators.

This module provides the routines for upscaling_operators.

Authors

Giovanni Dalmasso, Rohini Kumar

Date

Dec 2012

15.98.2 Function/Subroutine Documentation

15.98.2.1 l0_fractionalcover_in_lx()

```
real(dp) function, dimension(size(l0upbound_inlx, 1)), public mo_upscaling_operators::l0_<-
fractionalcover_in_lx (
    integer(i4), dimension(:), intent(in) dataIn0,
    integer(i4), intent(in) classId,
    logical, dimension(:, :), intent(in) mask0,
    integer(i4), dimension(:), intent(in) L0upBound_inLx,
    integer(i4), dimension(:), intent(in) L0downBound_inLx,
    integer(i4), dimension(:), intent(in) L0leftBound_inLx,
    integer(i4), dimension(:), intent(in) L0rightBound_inLx,
    integer(i4), dimension(:), intent(in) nTCells0_inLx )
```

fractional coverage of a given class of L0 fields in Lx field (Lx = L1 or L11)

Fractional coverage of a given class of L0 fields in Lx field (Lx = L1 or L11). For example, this routine can be used for calculating the karstic fraction.

Parameters

in	integer(i4), dimension(:) :: dataIn0	input fields at finer scale
in	integer(i4) :: classId	class id for which fraction has to be estimated
in	logical, dimension(:, :) :: mask0	finer scale L0 mask
in	integer(i4), dimension(:) :: L0upBound_inLx	row start at finer L0 scale
in	integer(i4), dimension(:) :: L0downBound_inLx	row end at finer L0 scale
in	integer(i4), dimension(:) :: L0leftBound_inLx	col start at finer L0 scale
in	integer(i4), dimension(:) :: L0rightBound_inLx	col end at finer L0 scale
in	integer(i4), dimension(:) :: nTCells0_inLx	total number of valid L0 cells in a given Lx cell

Returns

real(dp) :: L0_fractionalCover_in_Lx(:) — packed 1D fraction coverage (Lx) of given class id

Authors

Rohini Kumar

Date

Feb 2013

References mo_common_constants::nodata_i4.

Referenced by mo_multi_param_reg::karstic_layer(), and mo_multi_param_reg::mpr().

Here is the caller graph for this function:



15.98.2.2 majority_statistics()

```

integer(i4) function, dimension(size(l1_upper_rowid_cell, 1)), public mo_upscaling_operators::>
::majority_statistics (
    integer(i4), intent(in) nClass,
    integer(i4), dimension(:), intent(in) L1_upper_rowId_cell,
    integer(i4), dimension(:), intent(in) L1_lower_rowId_cell,
    integer(i4), dimension(:), intent(in) L1_left_colonId_cell,
    integer(i4), dimension(:), intent(in) L1_right_colonId_cell,
    integer(i4), dimension(:, :), intent(in) L0_fineScale_2D_data )
  
```

majority statistics

upscale grid L0_fineScale_2D_data based on a majority statistics

Parameters

in	integer(i4) :: nClass	number of classes
in	integer(i4), dimension(:) :: L1_upper_rowId_cell	upper row boundary (level-0) of a level-1 cell
in	integer(i4), dimension(:) :: L1_lower_rowId_cell	lower row boundary (level-0) of a level-1 cell
in	integer(i4), dimension(:) :: L1_left_colonId_cell	left colon boundary (level-0) of a level-1 cell
in	integer(i4), dimension(:) :: L1_right_colonId_cell	right colon boundary (level-0) of a level-1 cell
in	integer(i4), dimension(:, :) :: L0_fineScale_2D_data	high resolution data

Returns

integer(i4) :: majority_statistics(:) — Upscaled variable based on majority.

Authors

Giovanni Dalmasso, Rohini Kumar

Date

Dec 2012

15.98.2.3 upscale_arithmetic_mean()

```
real(dp) function, dimension(size(nL0_cells_in_L1_cell, 1)), public mo_upscaling_operators<-
::upscale_arithmetic_mean (
    integer(i4), dimension(:), intent(in) nL0_cells_in_L1_cell,
    integer(i4), dimension(:), intent(in) L1_upper_rowId_cell,
    integer(i4), dimension(:), intent(in) L1_lower_rowId_cell,
    integer(i4), dimension(:), intent(in) L1_left_colonId_cell,
    integer(i4), dimension(:), intent(in) L1_right_colonId_cell,
    integer(i4), dimension(:), intent(in) L0_cellId,
    logical, dimension(:, :), intent(in) mask0,
    real(dp), intent(in) nodata_value,
    real(dp), dimension(:), intent(in) L0_fineScale_data )
```

arithmetic mean

upscaling of level-0 grid data to level-1 using arithmetic mean

Parameters

in	integer(i4), dimension(:) :: nL0_cells_in_L1_cell	number of level-0 cells within a level-1 cell
in	integer(i4), dimension(:) :: L1_upper_rowId_cell	upper row boundary (level-0) of a level-1 cell
in	integer(i4), dimension(:) :: L1_lower_rowId_cell	lower row boundary (level-0) of a level-1 cell
in	integer(i4), dimension(:) :: L1_left_colonId_cell	left colon boundary (level-0) of a level-1 cell
in	integer(i4), dimension(:) :: L1_right_colonId_cell	right colon boundary (level-0) of a level-1 cell
in	integer(i4), dimension(:) :: L0_cellId	cell ID at level-0
in	logical, dimension(:, :) :: mask0	mask at level 0
in	real(dp) :: nodata_value	no data value
in	real(dp), dimension(:) :: L0_fineScale_data	high resolution data

Returns

real(dp) :: upscale_arithmetic_mean() — Upscaled variable from L0 to L1 using arithmetic mean

Authors

Giovanni Dalmasso, Rohini Kumar

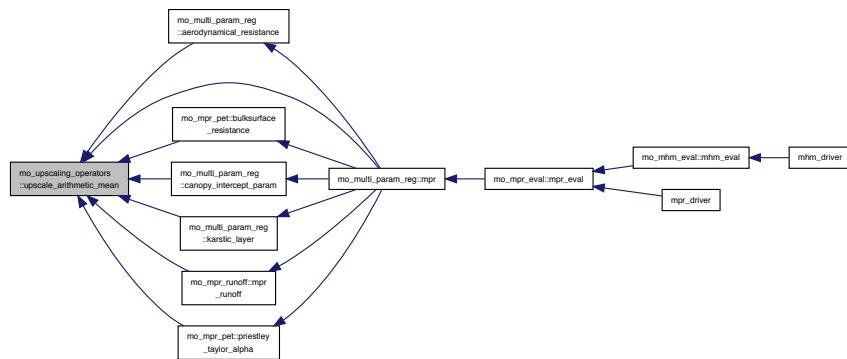
Date

Dec 2012

References mo_kind::i4.

Referenced by mo_multi_param_reg::aerodynamical_resistance(), mo_mpr_pet::bulksurface_resistance(), mo_multi_param_reg::canopy_intercept_param(), mo_multi_param_reg::karstic_layer(), mo_multi_param_reg::mpr(), mo_mpr_runoff::mpr_runoff(), and mo_mpr_pet::priestley_taylor_alpha().

Here is the caller graph for this function:



15.98.2.4 upscale_geometric_mean()

```
real(dp) function, dimension(size(l1_upper_rowid_cell, 1)), public mo_upscaling_operators::upscale_geometric_mean (
    integer(i4), dimension(:), intent(in) l1_upper_rowId_cell,
    integer(i4), dimension(:), intent(in) l1_lower_rowId_cell,
    integer(i4), dimension(:), intent(in) l1_left_colonId_cell,
    integer(i4), dimension(:), intent(in) l1_right_colonId_cell,
    logical, dimension(:, :), intent(in) mask0,
    real(dp), intent(in) nodata_value,
    real(dp), dimension(:), intent(in) l0_fineScale_data )
```

geometric mean

upscaling of level-0 grid data to level-1 using geometric mean

Parameters

in	integer(i4), dimension(:) :: l1_upper_rowid_cell	upper row boundary (level-0) of a level-1 cell
in	integer(i4), dimension(:) :: l1_lower_rowid_cell	lower row boundary (level-0) of a level-1 cell
in	integer(i4), dimension(:) :: l1_left_colonid_cell	left colon boundary (level-0) of a level-1 cell
in	integer(i4), dimension(:) :: l1_right_colonid_cell	right colon boundary (level-0) of a level-1 cell
in	logical, dimension(:, :) :: mask0	mask at level 0
in	real(dp) :: nodata_value	no data value
in	real(dp), dimension(:) :: l0_fineScale_data	high resolution data

Returns

`real(dp) :: upscale_geometric_mean(:)` — Upscaled variable from L0 to L1 using geometric mean

Authors

Giovanni Dalmasso, Rohini Kumar

Date

Dec 2012

15.98.2.5 upscale_harmonic_mean()

```
real(dp) function, dimension(size(nL0_cells_in_L1_cell, 1)), public mo_upscaling_operators<-
::upscale_harmonic_mean (
    integer(i4), dimension(:), intent(in) nL0_cells_in_L1_cell,
    integer(i4), dimension(:), intent(in) L1_upper_rowId_cell,
    integer(i4), dimension(:), intent(in) L1_lower_rowId_cell,
    integer(i4), dimension(:), intent(in) L1_left_colonId_cell,
    integer(i4), dimension(:), intent(in) L1_right_colonId_cell,
    integer(i4), dimension(:), intent(in) L0_cellId,
    logical, dimension(:, :), intent(in) mask0,
    real(dp), intent(in) nodata_value,
    real(dp), dimension(:, ), intent(in) L0_fineScale_data )
```

harmonic mean

upscaling of level-0 grid data to level-1 using harmonic mean

Parameters

in	<code>integer(i4), dimension(:) :: nL0_cells_in_L1_cell</code>	number of level-0 cells within a level-1 cell
in	<code>integer(i4), dimension(:) :: L1_upper_rowId_cell</code>	upper row boundary (level-0) of a level-1 cell
in	<code>integer(i4), dimension(:) :: L1_lower_rowId_cell</code>	lower row boundary (level-0) of a level-1 cell
in	<code>integer(i4), dimension(:) :: L1_left_colonId_cell</code>	left colon boundary (level-0) of a level-1 cell
in	<code>integer(i4), dimension(:) :: L1_right_colonId_cell</code>	right colon boundary (level-0) of a level-1 cell
in	<code>integer(i4), dimension(:) :: L0_cellId</code>	cell ID at level-0
in	<code>logical, dimension(:, :) :: mask0</code>	mask at Level 0
in	<code>real(dp) :: nodata_value</code>	no data value
in	<code>real(dp), dimension(:,) :: L0_fineScale_data</code>	high resolution data

Returns

`real(dp) :: upscale_harmonic_mean(:)` — Upscaled variable from L0 to L1 using harmonic mean

Authors

Giovanni Dalmasso, Rohini Kumar

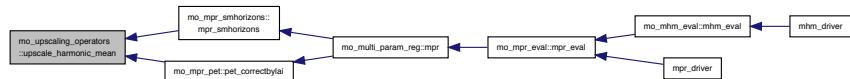
Date

Dec 2012

References mo_kind::i4.

Referenced by mo_mpr_smhorizons::mpr_smhorizons(), and mo_mpr_pet::pet_correctbylai().

Here is the caller graph for this function:



15.98.2.6 upscale_p_norm()

```

real(dp) function, dimension(size(nl0_cells_in_l1_cell, 1)) mo_upscaling_operators::upscale_p_norm (
    integer(i4), dimension(:), intent(in) nl0_cells_in_l1_cell,
    integer(i4), dimension(:), intent(in) l1_upper_rowId_cell,
    integer(i4), dimension(:), intent(in) l1_lower_rowId_cell,
    integer(i4), dimension(:), intent(in) l1_left_colonId_cell,
    integer(i4), dimension(:), intent(in) l1_right_colonId_cell,
    integer(i4), dimension(:), intent(in) l0_cellId,
    logical, dimension(:, :), intent(in) mask0,
    real(dp), intent(in) nodata_value,
    real(dp), intent(in) p_norm,
    real(dp), dimension(:, ), intent(in) l0_fineScale_data )

```

arithmetic mean

upscaling of level-0 grid data to level-1 using arithmetic mean

Parameters

in	integer(i4), dimension(:) :: nl0_cells_in_l1_cell	number of level-0 cells within a level-1 cell
in	integer(i4), dimension(:) :: l1_upper_rowId_cell	upper row boundary (level-0) of a level-1 cell
in	integer(i4), dimension(:) :: l1_lower_rowId_cell	lower row boundary (level-0) of a level-1 cell
in	integer(i4), dimension(:) :: l1_left_colonId_cell	left colon boundary (level-0) of a level-1 cell
in	integer(i4), dimension(:) :: l1_right_colonId_cell	right colon boundary (level-0) of a level-1 cell
in	integer(i4), dimension(:) :: l0_cellId	cell ID at level-0
in	logical, dimension(:, :) :: mask0	mask at level 0
in	real(dp) :: nodata_value	no data value
in	real(dp) :: p_norm	p_norm value
in	real(dp), dimension(:,) :: l0_fineScale_data	high resolution data

Returns

real(dp) :: upscale_arithmetic_mean() — Upscaled variable from L0 to L1 using arithmetic mean

Authors

Giovanni Dalmasso, Rohini Kumar

Date

Dec 2012

References mo_kind::i4.

15.99 mo_utils Module Reference

General utilities for the CHS library.

Data Types

- interface [eq](#)
- interface [equal](#)

Comparison of real values.

- interface [ge](#)
- interface [greaterequal](#)
- interface [is_finite](#)

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

- interface [is_nan](#)
- interface [is_normal](#)
- interface [le](#)
- interface [lesserequal](#)
- interface [locate](#)

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

- interface [ne](#)
- interface [notequal](#)
- interface [special_value](#)

Special IEEE values.

- interface [swap](#)

Swap to values or two elements in array.

Functions/Subroutines

- elemental pure logical function [equal_dp](#) (a, b)
- elemental pure logical function [equal_sp](#) (a, b)
- elemental pure logical function [greaterequal_dp](#) (a, b)
- elemental pure logical function [greaterequal_sp](#) (a, b)
- elemental pure logical function [lesserequal_dp](#) (a, b)
- elemental pure logical function [lesserequal_sp](#) (a, b)
- elemental pure logical function [notequal_dp](#) (a, b)
- elemental pure logical function [notequal_sp](#) (a, b)
- elemental pure logical function [is_finite_dp](#) (a)
- elemental pure logical function [is_finite_sp](#) (a)
- elemental pure logical function [is_nan_dp](#) (a)
- elemental pure logical function [is_nan_sp](#) (a)
- elemental pure logical function [is_normal_dp](#) (a)
- elemental pure logical function [is_normal_sp](#) (a)

- integer(i4) function `locate_0d_dp` (x, y)
- integer(i4) function `locate_0d_sp` (x, y)
- integer(i4) function, dimension(:), allocatable `locate_1d_dp` (x, y)
- integer(i4) function, dimension(:), allocatable `locate_1d_sp` (x, y)
- elemental pure subroutine `swap_xy_dp` (x, y)
- elemental pure subroutine `swap_xy_sp` (x, y)
- elemental pure subroutine `swap_xy_i4` (x, y)
- subroutine `swap_vec_dp` (x, i1, i2)
- subroutine `swap_vec_sp` (x, i1, i2)
- subroutine `swap_vec_i4` (x, i1, i2)
- real(dp) function `special_value_dp` (x, ieee)
- real(sp) function `special_value_sp` (x, ieee)

15.99.1 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

15.99.2 Function/Subroutine Documentation

15.99.2.1 equal_dp()

```
elemental pure logical function mo_utils::equal_dp (
    real(dp), intent(in) a,
    real(dp), intent(in) b )  [private]
```

15.99.2.2 equal_sp()

```
elemental pure logical function mo_utils::equal_sp (
    real(sp), intent(in) a,
    real(sp), intent(in) b )  [private]
```

15.99.2.3 greataorequal_dp()

```
elemental pure logical function mo_utils::greataorequal_dp (
    real(dp), intent(in) a,
    real(dp), intent(in) b )  [private]
```

15.99.2.4 `greaterequal_sp()`

```
elemental pure logical function mo_utils::greaterequal_sp (
    real(sp), intent(in) a,
    real(sp), intent(in) b ) [private]
```

15.99.2.5 `is_finite_dp()`

```
elemental pure logical function mo_utils::is_finite_dp (
    real(dp), intent(in) a ) [private]
```

15.99.2.6 `is_finite_sp()`

```
elemental pure logical function mo_utils::is_finite_sp (
    real(sp), intent(in) a ) [private]
```

15.99.2.7 `is_nan_dp()`

```
elemental pure logical function mo_utils::is_nan_dp (
    real(dp), intent(in) a ) [private]
```

15.99.2.8 `is_nan_sp()`

```
elemental pure logical function mo_utils::is_nan_sp (
    real(sp), intent(in) a ) [private]
```

15.99.2.9 `is_normal_dp()`

```
elemental pure logical function mo_utils::is_normal_dp (
    real(dp), intent(in) a ) [private]
```

15.99.2.10 `is_normal_sp()`

```
elemental pure logical function mo_utils::is_normal_sp (
    real(sp), intent(in) a ) [private]
```

15.99.2.11 `lesserequal_dp()`

```
elemental pure logical function mo_utils::lesserequal_dp (
    real(dp), intent(in) a,
    real(dp), intent(in) b ) [private]
```

15.99.2.12 lesserequal_sp()

```
elemental pure logical function mo_utils::lesserequal_sp (
    real(sp), intent(in) a,
    real(sp), intent(in) b )  [private]
```

15.99.2.13 locate_0d_dp()

```
integer(i4) function mo_utils::locate_0d_dp (
    real(dp), dimension(:), intent(in) x,
    real(dp), intent(in) y )  [private]
```

15.99.2.14 locate_0d_sp()

```
integer(i4) function mo_utils::locate_0d_sp (
    real(sp), dimension(:), intent(in) x,
    real(sp), intent(in) y )  [private]
```

15.99.2.15 locate_1d_dp()

```
integer(i4) function, dimension(:), allocatable mo_utils::locate_1d_dp (
    real(dp), dimension(:), intent(in) x,
    real(dp), dimension(:), intent(in) y )  [private]
```

15.99.2.16 locate_1d_sp()

```
integer(i4) function, dimension(:), allocatable mo_utils::locate_1d_sp (
    real(sp), dimension(:), intent(in) x,
    real(sp), dimension(:), intent(in) y )  [private]
```

15.99.2.17 notequal_dp()

```
elemental pure logical function mo_utils::notequal_dp (
    real(dp), intent(in) a,
    real(dp), intent(in) b )  [private]
```

15.99.2.18 notequal_sp()

```
elemental pure logical function mo_utils::notequal_sp (
    real(sp), intent(in) a,
    real(sp), intent(in) b )  [private]
```

15.99.2.19 special_value_dp()

```
real(dp) function mo_utils::special_value_dp (
    real(dp), intent(in) x,
    character(len = *), intent(in) ieee ) [private]
```

References mo_string_utils::toupper().

Here is the call graph for this function:



15.99.2.20 special_value_sp()

```
real(sp) function mo_utils::special_value_sp (
    real(sp), intent(in) x,
    character(len = *), intent(in) ieee ) [private]
```

References mo_string_utils::toupper().

Here is the call graph for this function:



15.99.2.21 swap_vec_dp()

```
subroutine mo_utils::swap_vec_dp (
    real(dp), dimension(:), intent(inout) x,
    integer(i4), intent(in) i1,
    integer(i4), intent(in) i2 ) [private]
```

15.99.2.22 swap_vec_i4()

```
subroutine mo_utils::swap_vec_i4 (
    integer(i4), dimension(:), intent(inout) x,
```

```
integer(i4), intent(in) i1,
integer(i4), intent(in) i2 ) [private]
```

15.99.2.23 swap_vec_sp()

```
subroutine mo_utils::swap_vec_sp (
    real(sp), dimension(:), intent(inout) x,
    integer(i4), intent(in) i1,
    integer(i4), intent(in) i2 ) [private]
```

15.99.2.24 swap_xy_dp()

```
elemental pure subroutine mo_utils::swap_xy_dp (
    real(dp), intent(inout) x,
    real(dp), intent(inout) y ) [private]
```

15.99.2.25 swap_xy_i4()

```
elemental pure subroutine mo_utils::swap_xy_i4 (
    integer(i4), intent(inout) x,
    integer(i4), intent(inout) y ) [private]
```

15.99.2.26 swap_xy_sp()

```
elemental pure subroutine mo_utils::swap_xy_sp (
    real(sp), intent(inout) x,
    real(sp), intent(inout) y ) [private]
```

15.100 mo_write_ascii Module Reference

Module to write ascii file output.

Functions/Subroutines

- subroutine, public [write_configfile](#)
This module 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.

15.100.1 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

15.100.2 Function/Subroutine Documentation

15.100.2.1 write_configfile()

```
subroutine, public mo_write_ascii::write_configfile ( )
```

This module writes the results of the configuration into an ASCII-file.

TODO: add description

Authors

Christoph Schneider

Date

May 2013 TODO: add description

TODO: add description

Authors

Robert Schweppe

Date

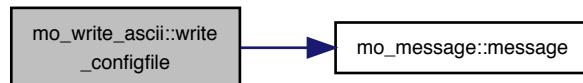
Jun 2018

References mo_common_variables::dirconfigout, mo_mrm_global_variables::dirgauges, mo_common_variables::dirlcover, mo_common_variables::dirmorpho, mo_common_variables::dirout, mo_global_variables::dirprecipitation, mo_global_variables::dirreferenceet, mo_common_variables::dirrestartout, mo_global_variables::dirtemperature, mo_common_mhm_mrm_variables::evalper, mo_common_file::file_config, mo_mrm_global_variables::gauge, mo_common_variables::global_parameters, mo_common_variables::global_parameters_name, mo_kind::i4, mo_common_variables::iflag_coordinate_sys, mo_mrm_global_variables::inflowgauge, mo_common_variables::i0_basin, mo_mrm_global_variables::i11_fromn, mo_mrm_global_variables::i11_label, mo_mrm_global_variables::i11_length, mo_mrm_global_variables::i11_netperm, mo_mrm_global_variables::i11_rorder, mo_mrm_global_variables::i11_slope, mo_mrm_global_variables::i11_ton, mo_mrm_global_variables::i1_i11_id, mo_common_variables::lc_year_end, mo_common_variables::lc_year_start, mo_common_variables::lcfilename, mo_common_mhm_mrm_variables::lcyearid, mo_common_variables::level0, mo_common_variables::level1, mo_mrm_global_variables::level11, mo_message::message(), mo_common_variables::nbasins, mo_mrm_global_variables::ngaugestotal, mo_mrm_global_variables::ninfilegaugestotal, mo_common_variables::nlcoverscene, mo_common_constants::nodata_dp, mo_common_variables::processmatrix,

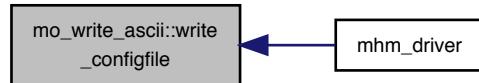
mo_common_mhm_mrm_variables::read_restart, mo_common_variables::resolutionhydrology, mo_common_mhm_mrm_variables::resolutionrouting, mo_common_mhm_mrm_variables::simper, mo_common_mhm_mrm_variables::timestep, mo_common_file::uconfig, mo_file::version, mo_common_mhm_mrm_variables::warmper, and mo_common_variables::write_restart.

Referenced by mhm_driver().

Here is the call graph for this function:



Here is the caller graph for this function:



15.100.2.2 write_optifile()

```

subroutine, public mo_write_ascii::write_optifile (
    real(dp), intent(in) best_OF,
    real(dp), dimension(:), intent(in) best_paramSet,
    character(len = *), dimension(:), intent(in) param_names )

```

Write briefly final optimization results.

Write overall best objective function and the best optimized parameter set to a file_opti.

Parameters

in	<i>real(dp) :: best_OF</i>	best objective function value as returned by the optimization routine
in	<i>real(dp), dimension(:) :: best_paramSet</i>	best associated global parameter set Called only when optimize is .TRUE.
in	<i>character(len = *), dimension(:) :: param_names</i>	

Authors

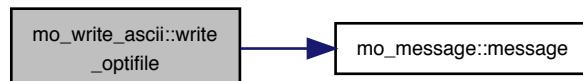
David Schaefer

Date

July 2013

References `mo_common_variables::dirconfigout`, `mo_common_mhm_mrm_file::file_opti`, `mo_message::message()`, and `mo_common_mhm_mrm_file::uopti`.

Here is the call graph for this function:

15.100.2.3 `write_optinamelist()`

```

subroutine, public mo_write_ascii::write_optinamelist (
    integer(i4), dimension(nprocesses, 3), intent(in) processMatrix,
    real(dp), dimension(:, :, ), intent(in) parameters,
    logical, dimension(size(parameters, 1)), intent(in) maskpara,
    character(len = *), dimension(size(parameters, 1)), intent(in) parameters_name )

```

Write final, optimized parameter set in a namelist format.

Write final, optimized parameter set in a namelist format. Only parameters of processes which were switched on are written to the namelist. All others are discarded.

Parameters

in	<code>integer(i4), dimension(nProcesses, 3) :: processMatrix</code>	information about which process case was used
in	<code>real(dp), dimension(:, :,) :: parameters</code>	(min, max, opti)
in	<code>logical, dimension(size(parameters, 1)) :: maskpara</code>	.true. if parameter was calibrated
in	<code>character(len = *), dimension(size(parameters, 1)) :: parameters_name</code>	clear names of parameters

Authors

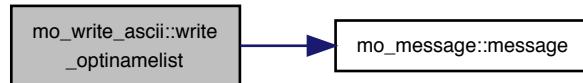
Juliane Mai

Date

Dec 2013

References `mo_common_variables::dirconfigout`, `mo_common_mhm_mrm_file::file_opti_nml`, `mo_message::message()`, `mo_common_variables::nprocesses`, and `mo_common_mhm_mrm_file::uopti_nml`.

Here is the call graph for this function:



15.101 mo_write_fluxes_states Module Reference

Creates NetCDF output for different fluxes and state variables of mHM.

Data Types

- interface `outputdataset`
- interface `outputvariable`

Functions/Subroutines

- type(`outputvariable`) function `newoutputvariable` (nc, name, dtype, dims, ncells, mask, avg)
Initialize OutputVariable.
- subroutine `updatevariable` (self, data)
Update OutputVariable.
- subroutine `writevariabletimestep` (self, timestep)
Write timestep to file.
- type(`outputdataset`) function, public `newoutputdataset` (ibasin, mask1, nCells)
Initialize OutputDataset.
- subroutine `updatedataset` (self, sidx, eidx, L1_fSealed, L1_fNotSealed, L1_inter, L1_snowPack, L1_soilMoist, L1_soilMoistSat, L1_sealSTW, L1_unsatSTW, L1_satSTW, L1_neutrons, L1_pet, L1_aETSoil, L1_aETCanopy, L1_aETSealed, L1_total_runoff, L1_runoffSeal, L1_fastRunoff, L1_slowRunoff, L1_baseflow, L1_percol, L1_infilSoil, L1_preEffect)
Update all variables.
- subroutine `writetimestep` (self, timestep)
Write all accumulated data.
- subroutine `close` (self)
Close the file.
- type(`ncdataset`) function `createoutputfile` (ibasin)
Create and initialize output file.
- subroutine `writevariableattributes` (var, long_name, unit)
Write output variable attributes.
- character(16) function `fluxesunit` (ibasin)
Generate a unit string.

15.101.1 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

15.101.2 Function/Subroutine Documentation

15.101.2.1 close()

```
subroutine mo_write_fluxes_states::close (
    class(outputdataset) self ) [private]
```

Close the file.

Close the file associated with variable of type(OutputDataset)

Authors

Rohini Kumar & Stephan Thober

Date

August 2013

References mo_common_variables::dirout, and mo_message::message().

Here is the call graph for this function:



15.101.2.2 createoutputfile()

```
type(ncdataset) function mo_write_fluxes_states::createoutputfile (
    integer(i4), intent(in) ibasin )
```

Create and initialize output file.

Create output file, write all non-dynamic variables and global attributes for the given basin.

Returns

type(NcDataset)

Parameters

in	integer(i4) :: <i>ibasin</i>	-> basin id
----	------------------------------	-------------

Authors

David Schaefer

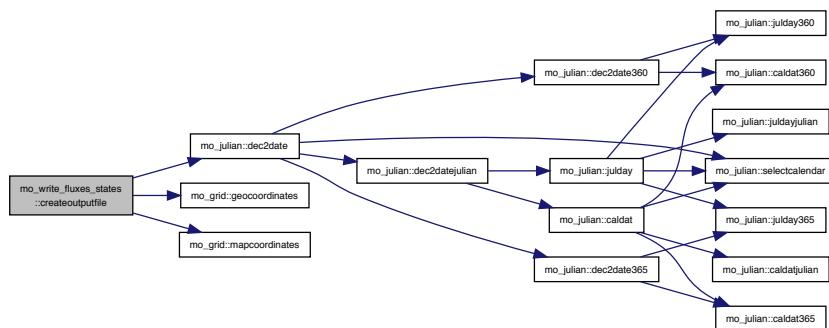
Date

June 2015

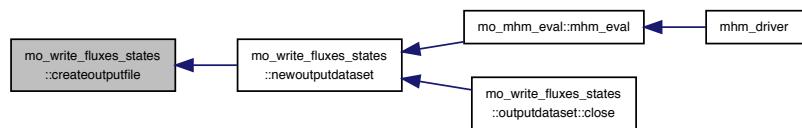
References `mo_julian::dec2date()`, `mo_common_variables::dirout`, `mo_common_mhm_mrm_variables::evalper`, `mo_grid::geocoordinates()`, `mo_common_variables::level1`, `mo_grid::mapcoordinates()`, `mo_common_constants::nodata_dp`, and `mo_file::version`.

Referenced by `newoutputdataset()`.

Here is the call graph for this function:



Here is the caller graph for this function:



15.101.2.3 fluxesunit()

```
character(16) function mo_write_fluxes_states::fluxesunit (
    integer(i4), intent(in) ibasin ) [private]
```

Generate a unit string.

Generate the unit string for the output variable netcdf attribute based on modeling timestep

Returns

character(16)

Parameters

in	integer(i4) :: <i>ibasin</i>	
----	------------------------------	--

Authors

David Schaefer

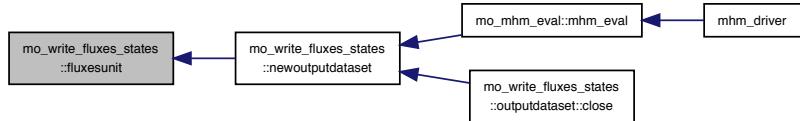
Date

June 2015

References `mo_common_mhm_mrm_variables::ntstepday`, `mo_common_mhm_mrm_variables::simper`, `mo_common_mhm_mrm_variables::timestep`, and `mo_global_variables::timestep_model_outputs`.

Referenced by `newoutputdataset()`.

Here is the caller graph for this function:



15.101.2.4 newoutputdataset()

```
type(outputdataset) function, public mo_write_fluxes_states::newoutputdataset (
    integer(i4), intent(in) ibasin,
    logical, dimension(:, :), intent(in), target 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) :: <i>ibasin</i>	-> basin id
in	logical, dimension(:, :) :: <i>mask1</i>	-> L1 mask to reconstruct the data
in	integer(i4) :: <i>nCells</i>	

Authors

Matthias Zink

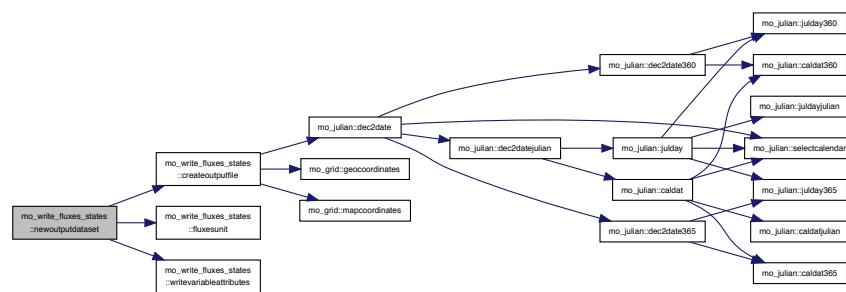
Date

Apr 2013

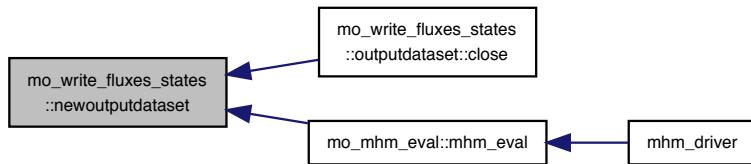
References `createoutputfile()`, `fluxesunit()`, `mo_mpr_global_variables::nsoilhorizons_mhm`, `mo_global_variables::outputflxstate`, and `writevariableattributes()`.

Referenced by `mo_write_fluxes_states::outputdataset::close()`, and `mo_mhm_eval::mhm_eval()`.

Here is the call graph for this function:



Here is the caller graph for this function:



15.101.2.5 newoutputvariable()

```

type(outputvariable) function mo_write_fluxes_states::newoutputvariable (
    type(ncdataset), intent(in) nc,
    character(*), intent(in) name,
    character(*), intent(in) dtype,
    character(16), dimension(3), intent(in) dims,
    integer(i4), intent(in) ncells,
    logical, dimension(:, :, ), intent(in), target mask,
    logical, intent(in), optional avg ) [private]

```

Initialize OutputVariable.

TODO: add description

Returns

```
type(OutputVariable)
```

Parameters

in	<i>type(NcDataset) :: nc</i>	-> NcDataset which contains the variable
in	<i>character(*) :: name</i>	
in	<i>character(*) :: dtype</i>	
in	<i>character(16), dimension(3) :: dims</i>	
in	<i>integer(i4) :: ncells</i>	-> number of cells in basin
in	<i>logical, dimension(:, :) :: mask</i>	
in	<i>logical, optional :: avg</i>	-> average the data before writing

Authors

David Schaefer

Date

June 2015

15.101.2.6 updatedataset()

```
subroutine mo_write_fluxes_states::updatedataset (
    class(outputdataset), intent(inout), target self,
    integer(i4), intent(in) sidx,
    integer(i4), intent(in) eidx,
    real(dp), dimension(:, ), intent(in) L1_fSealed,
    real(dp), dimension(:, ), intent(in) L1_fNotSealed,
    real(dp), dimension(:, ), intent(in) L1_inter,
    real(dp), dimension(:, ), intent(in) L1_snowPack,
    real(dp), dimension(:, :, ), intent(in) L1_soilMoist,
    real(dp), dimension(:, :, ), intent(in) L1_soilMoistSat,
    real(dp), dimension(:, ), intent(in) L1_sealSTW,
    real(dp), dimension(:, ), intent(in) L1_unsatSTW,
    real(dp), dimension(:, ), intent(in) L1_satSTW,
    real(dp), dimension(:, ), intent(in) L1_neutrons,
    real(dp), dimension(:, ), intent(in) L1_pet,
    real(dp), dimension(:, :, ), intent(in) L1_aETSoil,
    real(dp), dimension(:, ), intent(in) L1_aETCanopy,
    real(dp), dimension(:, ), intent(in) L1_aETSealed,
    real(dp), dimension(:, ), intent(in) L1_total_runoff,
    real(dp), dimension(:, ), intent(in) L1_runoffSeal,
    real(dp), dimension(:, ), intent(in) L1_fastRunoff,
    real(dp), dimension(:, ), intent(in) L1_slowRunoff,
    real(dp), dimension(:, ), intent(in) L1_baseflow,
    real(dp), dimension(:, ), intent(in) L1_percol,
    real(dp), dimension(:, :, ), intent(in) L1_infilSoil,
    real(dp), dimension(:, ), intent(in) L1_preEffect )
```

Update all variables.

Call the type bound procedure updateVariable for all output variables. If a new output variable needs to be written, this is the second of two procedures to change (first: newOutputDataset)

Parameters

in, out	<i>class(OutputDataset) :: self</i>	
in	<i>integer(i4) :: sidx, eidx</i>	
in	<i>integer(i4) :: sidx, eidx</i>	
in	<i>real(dp), dimension(:) :: L1_fSealed</i>	
in	<i>real(dp), dimension(:) :: L1_fNotSealed</i>	
in	<i>real(dp), dimension(:) :: L1_inter</i>	
in	<i>real(dp), dimension(:) :: L1_snowPack</i>	
in	<i>real(dp), dimension(:, :) :: L1_soilMoist</i>	
in	<i>real(dp), dimension(:, :) :: L1_soilMoistSat</i>	
in	<i>real(dp), dimension(:) :: L1_sealSTW</i>	
in	<i>real(dp), dimension(:) :: L1_unsatSTW</i>	
in	<i>real(dp), dimension(:) :: L1_satSTW</i>	
in	<i>real(dp), dimension(:) :: L1_neutrons</i>	
in	<i>real(dp), dimension(:) :: L1_pet</i>	
in	<i>real(dp), dimension(:, :) :: L1_aETSoil</i>	
in	<i>real(dp), dimension(:) :: L1_aETCanopy</i>	
in	<i>real(dp), dimension(:) :: L1_aETSealed</i>	
in	<i>real(dp), dimension(:) :: L1_total_runoff</i>	
in	<i>real(dp), dimension(:) :: L1_runoffSeal</i>	
in	<i>real(dp), dimension(:) :: L1_fastRunoff</i>	
in	<i>real(dp), dimension(:) :: L1_slowRunoff</i>	
in	<i>real(dp), dimension(:) :: L1_baseflow</i>	
in	<i>real(dp), dimension(:) :: L1_percol</i>	
in	<i>real(dp), dimension(:, :) :: L1_infilSoil</i>	
in	<i>real(dp), dimension(:) :: L1_preEffect</i>	

Authors

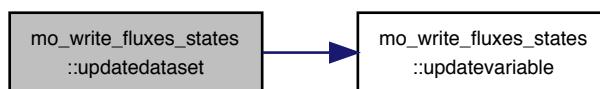
Matthias Zink

Date

Apr 2013

References `mo_mpr_global_variables::nsoilhorizons_mhm`, `mo_global_variables::outputflxstate`, and `updatevariable()`.

Here is the call graph for this function:



15.101.2.7 updatevariable()

```
subroutine mo_write_fluxes_states::updatevariable (
    class(outputvariable), intent(inout) self,
    real(dp), dimension(:), intent(in) data ) [private]
```

Update OutputVariable.

Add the array given as actual argument to the derived type's component 'data'

Returns

type(OutputVariable)

Parameters

in, out	<i>class(OutputVariable) :: self</i>	
in	<i>real(dp), dimension(:) :: data</i>	

Authors

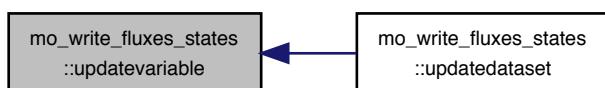
David Schaefer

Date

June 2015

Referenced by updatedataset().

Here is the caller graph for this function:



15.101.2.8 writetimestep()

```
subroutine mo_write_fluxes_states::writetimestep (
    class(outputdataset), intent(inout), target self,
    integer(i4), intent(in) timestep )
```

Write all accumulated data.

Write all accumulated and potentially averaged data to disk.

Returns

type(OutputVariable)

Parameters

in, out	<i>class(OutputDataset) :: self</i>	
in	<i>integer(i4) :: timestep</i>	The model timestep to write

Authors

David Schaefer

Date

June 2015

15.101.2.9 writevariableattributes()

```
subroutine mo_write_fluxes_states::writevariableattributes (
    type(outputvariable), intent(in) var,
    character(*), intent(in) long_name,
    character(*), intent(in) unit )
```

Write output variable attributes.

TODO: add description

Parameters

in	<i>type(OutputVariable) :: var</i>	
in	<i>character(*) :: long_name, unit</i>	-> variable name
in	<i>character(*) :: long_name, unit</i>	-> physical unit

Authors

David Schaefer

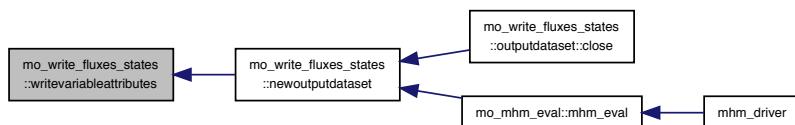
Date

June 2015

References mo_common_constants::nodata_dp.

Referenced by newoutputdataset().

Here is the caller graph for this function:



15.101.2.10 writevariabletimestep()

```
subroutine mo_write_fluxes_states::writevariabletimestep (
    class(outputvariable), intent(inout) self,
    integer(i4), intent(in) timestep ) [private]
```

Write timestep to file.

Write the content of the derived types's component 'data' to file, average if necessary

Parameters

in, out	<i>class(OutputVariable) :: self</i>	
in	<i>integer(i4) :: timestep</i>	-> index along the time dimension of the netcdf variable

Authors

David Schafer

Date

June 2015

References mo_common_constants::nodata_dp.

15.102 mo_xor4096 Module Reference

Data Types

- interface [get_timeseed](#)
- interface [xor4096](#)
- interface [xor4096g](#)

Functions/Subroutines

- subroutine [get_timeseed_i4_0d](#) (seed)
- subroutine [get_timeseed_i4_1d](#) (seed)
- subroutine [get_timeseed_i8_0d](#) (seed)
- subroutine [get_timeseed_i8_1d](#) (seed)
- subroutine [xor4096s_0d](#) (seed, SingleIntegerRN, save_state)
- subroutine [xor4096s_1d](#) (seed, SingleIntegerRN, save_state)
- subroutine [xor4096f_0d](#) (seed, SingleRealRN, save_state)
- subroutine [xor4096f_1d](#) (seed, SingleRealRN, save_state)
- subroutine [xor4096l_0d](#) (seed, DoubleIntegerRN, save_state)
- subroutine [xor4096l_1d](#) (seed, DoubleIntegerRN, save_state)
- subroutine [xor4096d_0d](#) (seed, DoubleRealRN, save_state)
- subroutine [xor4096d_1d](#) (seed, DoubleRealRN, save_state)
- subroutine [xor4096gf_0d](#) (seed, SingleRealRN, save_state)
- subroutine [xor4096gf_1d](#) (seed, SingleRealRN, save_state)
- subroutine [xor4096gd_0d](#) (seed, DoubleRealRN, save_state)
- subroutine [xor4096gd_1d](#) (seed, DoubleRealRN, save_state)

Variables

- `integer(i4), parameter, public n_save_state = 132_i4`
Dimension of vector saving the state of a stream.

15.102.1 Function/Subroutine Documentation

15.102.1.1 get_timeseed_i4_0d()

```
subroutine mo_xor4096::get_timeseed_i4_0d (
    integer(i4), intent(inout) seed )  [private]
```

15.102.1.2 get_timeseed_i4_1d()

```
subroutine mo_xor4096::get_timeseed_i4_1d (
    integer(i4), dimension(:), intent(inout) seed )  [private]
```

15.102.1.3 get_timeseed_i8_0d()

```
subroutine mo_xor4096::get_timeseed_i8_0d (
    integer(i8), intent(inout) seed )  [private]
```

References `mo_kind::i8`.

15.102.1.4 get_timeseed_i8_1d()

```
subroutine mo_xor4096::get_timeseed_i8_1d (
    integer(i8), dimension(:), intent(inout) seed )  [private]
```

References `mo_kind::i8`.

15.102.1.5 xor4096d_0d()

```
subroutine mo_xor4096::xor4096d_0d (
    integer(i8), intent(in) seed,
    real(dp), intent(out) DoubleRealRN,
    integer(i8), dimension(n_save_state), intent(inout), optional save_state )  [private]
```

References `n_save_state`.

15.102.1.6 xor4096d_1d()

```
subroutine mo_xor4096::xor4096d_1d (
    integer(i8), dimension(:), intent(in) seed,
    real(dp), dimension(size(seed, 1)), intent(out) DoubleRealRN,
```

```
    integer(i8), dimension(size(seed, 1), n_save_state), intent(inout), optional
  save_state ) [private]
```

References **n_save_state**.

15.102.1.7 xor4096f_0d()

```
subroutine mo_xor4096::xor4096f_0d (
    integer(i4), intent(in) seed,
    real(sp), intent(out) SingleRealRN,
    integer(i4), dimension(n_save_state), intent(inout), optional save_state ) [private]
```

References **n_save_state**.

15.102.1.8 xor4096f_1d()

```
subroutine mo_xor4096::xor4096f_1d (
    integer(i4), dimension(:, intent(in) seed,
    real(sp), dimension(size(seed)), intent(out) SingleRealRN,
    integer(i4), dimension(size(seed, 1), n_save_state), intent(inout), optional
  save_state ) [private]
```

References **n_save_state**.

15.102.1.9 xor4096gd_0d()

```
subroutine mo_xor4096::xor4096gd_0d (
    integer(i8), intent(in) seed,
    real(dp), intent(out) DoubleRealRN,
    integer(i8), dimension(n_save_state), intent(inout), optional save_state ) [private]
```

References **n_save_state**.

15.102.1.10 xor4096gd_1d()

```
subroutine mo_xor4096::xor4096gd_1d (
    integer(i8), dimension(:, intent(in) seed,
    real(dp), dimension(size(seed)), intent(out) DoubleRealRN,
    integer(i8), dimension(size(seed), n_save_state), intent(inout), optional save_←
  state ) [private]
```

References **n_save_state**.

15.102.1.11 xor4096gf_0d()

```
subroutine mo_xor4096::xor4096gf_0d (
    integer(i4), intent(in) seed,
    real(sp), intent(out) SingleRealRN,
    integer(i4), dimension(n_save_state), intent(inout), optional save_state ) [private]
```

References `n_save_state`.

15.102.1.12 xor4096gf_1d()

```
subroutine mo_xor4096::xor4096gf_1d (
    integer(i4), dimension(:), intent(in) seed,
    real(sp), dimension(size(seed)), intent(out) SingleRealRN,
    integer(i4), dimension(size(seed), n_save_state), intent(inout), optional save←
state ) [private]
```

References `n_save_state`.

15.102.1.13 xor4096l_0d()

```
subroutine mo_xor4096::xor4096l_0d (
    integer(i8), intent(in) seed,
    integer(i8), intent(out) DoubleIntegerRN,
    integer(i8), dimension(n_save_state), intent(inout), optional save_state ) [private]
```

References `n_save_state`.

15.102.1.14 xor4096l_1d()

```
subroutine mo_xor4096::xor4096l_1d (
    integer(i8), dimension(:), intent(in) seed,
    integer(i8), dimension(size(seed, 1)), intent(out) DoubleIntegerRN,
    integer(i8), dimension(size(seed, 1), n_save_state), intent(inout), optional
save_state ) [private]
```

References `n_save_state`.

15.102.1.15 xor4096s_0d()

```
subroutine mo_xor4096::xor4096s_0d (
    integer(i4), intent(in) seed,
    integer(i4), intent(out) SingleIntegerRN,
    integer(i4), dimension(n_save_state), intent(inout), optional save_state ) [private]
```

References `n_save_state`.

15.102.1.16 xor4096s_1d()

```
subroutine mo_xor4096::xor4096s_1d (
    integer(i4), dimension(:), intent(in) seed,
    integer(i4), dimension(size(seed, 1)), intent(out) SingleIntegerRN,
    integer(i4), dimension(size(seed, 1), n_save_state), intent(inout), optional
save_state ) [private]
```

References `n_save_state`.

15.102.2 Variable Documentation

15.102.2.1 n_save_state

```
integer(i4), parameter, public mo_xor4096::n_save_state = 132_i4
```

Dimension of vector saving the state of a stream.

Referenced by mo_sce::cce(), mo_sce::getpnt(), mo_mcmc::mcmc_dp(), mo_mcmc::mcmc_stddev_dp(), mo_sce::sce(), xor4096d_0d(), xor4096d_1d(), xor4096f_0d(), xor4096f_1d(), xor4096gd_0d(), xor4096gd_1d(), xor4096gf_0d(), xor4096gf_1d(), xor4096l_0d(), xor4096l_1d(), xor4096s_0d(), and xor4096s_1d().

Chapter 16

Data Type Documentation

16.1 mo_moment::absdev Interface Reference

Public Member Functions

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

16.1.1 Member Function/Subroutine Documentation

16.1.1.1 [absdev_dp\(\)](#)

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

16.1.1.2 [absdev_sp\(\)](#)

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

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

- [mo_moment.f90](#)

16.2 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, nLTERmax, Len, nST, eps, acc, seeds, printflag, maskpara, weight, changeParaMode, reflectionFlag, pertubFlexFlag, maxit, undef_funcval, tmp_file, funcbest, history)

16.2.1 Detailed Description

anneal

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

Parameters

in	<i>INTERFACE :: cost_dp</i>	interface calculating the cost function at a given point
in	<i>REAL(DP), DIMENSION(:) :: para</i>	initial parameter set
in	<i>REAL(DP), DIMENSION(size(para),2), optional :: prange</i>	lower and upper bound per parameter
in	<i>INTERFACE, optional :: prange_func</i>	interface calculating the feasible range for a parameter at a certain point, if ranges are variable
in	<i>REAL(DP), optional :: temp</i>	initial temperature DEFAULT: Get_Temperature
in	<i>REAL(DP), optional :: DT</i>	geometrical decreement of temperature $0.7 < DT < 0.999$ DEFAULT: 0.9_dp
in	<i>INTEGER(I4), optional :: nITERmax</i>	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	<i>INTEGER(I4), optional :: LEN</i>	Length of Markov Chain DEFAULT: MAX(250_i4, size(para,1))
in	<i>INTEGER(I4), optional :: nST</i>	Number of consecutive LEN steps DEFAULT: 5_i4
in	<i>REAL(DP), optional :: eps</i>	Stopping criteria of epsilon decreement of cost function DEFAULT: 0.0001_dp
in	<i>REAL(DP), optional :: acc</i>	Stopping criteria for Acceptance Ratio acc $\leq 0.1_dp$ DEFAULT: 0.1_dp
in	<i>INTEGER(I4/I8), DIMENSION(3), optional :: seeds</i>	Seeds of random numbers used for random parameter set generation DEFAULT: dependent on current time
in	<i>LOGICAL, optional :: printflag</i>	If .true. detailed command line output is written DEFAULT: .false.
in	<i>LOGICAL, DIMENSION(size(para)), optional :: maskpara</i>	maskpara(i) = .true. \rightarrow parameter is optimized maskpara(i) = .false. \rightarrow parameter is discarded from optimiztaion DEFAULT: .true.
in	<i>REAL(DP), DIMENSION(size(para)), optional :: weight</i>	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] \rightarrow parameter 2 is chosen twice as often as parameter 1 and 2 DEFAULT: weight = 1.0_dp
in	<i>INTEGER(I4), optional :: changeParaMode</i>	which and how many param. are changed in one step 1 = one parameter 2 = all parameter 3 = neighborhood parameter DEFAULT: 1_i4

Parameters

in	<i>LOGICAL, optional :: reflectionFlag</i>	if new parameter values are Gaussian distributed and reflected (.true.) or uniform in range (.false.) DEFAULT: .false.
in	<i>LOGICAL, optional :: pertubFlexFlag</i>	if perturbation of Gaussian distributed parameter values is constant at 0.2 (.false.) or depends on dR (.true.) DEFAULT: .true.
in	<i>LOGICAL, optional :: maxit</i>	maximizing (.true.) or minimizing (.false.) a function DEFAULT: .false. (minimization)
in	<i>REAL(DP), optional :: undef_funcval</i>	objective function value defining invalid model output, e.g. -999.0_dp
in	<i>character(len=*) , optional :: tmp_file</i>	file with temporal output
out	<i>REAL(DP), optional :: funcbest</i>	minimized value of cost function
out	<i>"REAL(DP),DIMENSION(:,),ALLOCATABLE,LE,optional</i>	:: 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.

16.2.2 Member Function/Subroutine Documentation

16.2.2.1 anneal_dp()

```
real(dp) function, dimension(size(para, 1)) mo_anneal::anneal::anneal_dp (
    procedure(eval_interface), intent(in), pointer eval,
    procedure(objective_interface), intent(in), pointer cost,
    real(dp), dimension(:), intent(in) para,
    real(dp), dimension(size(para, 1), 2), intent(in), optional prange,
    optional prange_func,
    real(dp), intent(in), optional temp,
    real(dp), intent(in), optional Dt,
    integer(i4), intent(in), optional nITERmax,
    integer(i4), intent(in), optional Len,
    integer(i4), intent(in), optional nST,
    real(dp), intent(in), optional eps,
    real(dp), intent(in), optional acc,
    integer(i8), dimension(3), intent(in), optional seeds,
```

```

logical, intent(in), optional printflag,
logical, dimension(size(para, 1)), intent(in), optional maskpara,
real(dp), dimension(size(para, 1)), intent(in), optional weight,
integer(i4), intent(in), optional changeParaMode,
logical, intent(in), optional reflectionFlag,
logical, intent(in), optional pertubFlexFlag,
logical, intent(in), optional maxit,
real(dp), intent(in), optional undef_funcval,
character(len = *), intent(in), optional tmp_file,
real(dp), intent(out), optional funcbest,
real(dp), dimension(:, :), intent(out), optional, allocatable history )

```

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

- [mo_anneal.f90](#)

16.3 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)

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

Author

Juliane Mai

Date

Aug 2012

16.3.2 Member Function/Subroutine Documentation

16.3.2.1 **append_char_3d()**

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

16.3.2.2 **append_char_m_m()**

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

16.3.2.3 **append_char_v_s()**

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

16.3.2.4 **append_char_v_v()**

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

16.3.2.5 append_dp_3d()

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

16.3.2.6 append_dp_m_m()

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

16.3.2.7 append_dp_v_s()

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

16.3.2.8 append_dp_v_v()

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

16.3.2.9 append_i4_m_m()

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

16.3.2.10 append_i4_v_s()

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

16.3.2.11 append_i4_v_v()

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

16.3.2.12 append_i8_3d()

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

16.3.2.13 append_i8_m_m()

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

16.3.2.14 append_i8_v_s()

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

16.3.2.15 append_i8_v_v()

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

16.3.2.16 append_lgt_3d()

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

16.3.2.17 append_lgt_m_m()

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

16.3.2.18 append_lgt_v_s()

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

16.3.2.19 append_lgt_v_v()

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

16.3.2.20 append_sp_3d()

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

16.3.2.21 append_sp_m_m()

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

16.3.2.22 append_sp_v_s()

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

16.3.2.23 append_sp_v_v()

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

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

- [mo_append.f90](#)

16.4 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)

16.4.1 Member Function/Subroutine Documentation

16.4.1.1 [arth_dp\(\)](#)

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

16.4.1.2 [arth_i4\(\)](#)

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

16.4.1.3 [arth_sp\(\)](#)

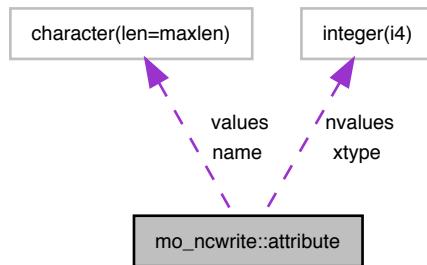
```
real(sp) function, dimension(n) mo_corr::arth::arth_sp (
    real(sp), intent(in) first,
    real(sp), intent(in) increment,
    integer(i4), intent(in) n ) [private]
```

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

- [mo_corr.f90](#)

16.5 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**

16.5.1 Member Data Documentation

16.5.1.1 name

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

16.5.1.2 nvalues

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

16.5.1.3 values

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

16.5.1.4 xtype

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

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

- [mo_ncwrite.f90](#)

16.6 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)

16.6.1 Member Function/Subroutine Documentation

16.6.1.1 [autocoeffk_1d_dp\(\)](#)

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

16.6.1.2 [autocoeffk_1d_sp\(\)](#)

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

16.6.1.3 [autocoeffk_dp\(\)](#)

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

16.6.1.4 [autocoeffk_sp\(\)](#)

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

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

- [mo_corr.f90](#)

16.7 `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`)

16.7.1 Member Function/Subroutine Documentation

16.7.1.1 `autocorr_1d_dp()`

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

16.7.1.2 `autocorr_1d_sp()`

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

16.7.1.3 `autocorr_dp()`

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

16.7.1.4 `autocorr_sp()`

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

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

- [mo_corr.f90](#)

16.8 `mo_moment::average` Interface Reference

Public Member Functions

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

16.8.1 Member Function/Subroutine Documentation

16.8.1.1 [average_dp\(\)](#)

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

16.8.1.2 [average_sp\(\)](#)

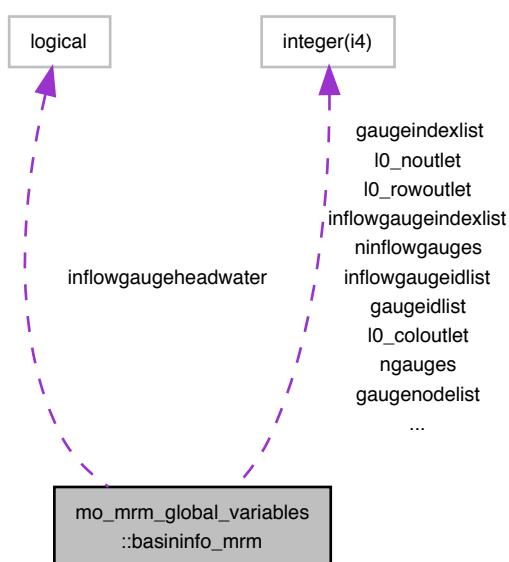
```
real(sp) function mo_moment::average::average_sp (
    real(sp), dimension(:), intent(in) dat,
    logical, dimension(:), intent(in), optional mask )
```

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

- [mo_moment.f90](#)

16.9 [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**

16.9.1 Member Data Documentation

16.9.1.1 gaugeidlist

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

16.9.1.2 gaugeindexlist

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

16.9.1.3 gaugenodelist

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

16.9.1.4 inflowgaugeheadwater

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

16.9.1.5 inflowgaugeidlist

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

16.9.1.6 inflowgaugeindexlist

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

16.9.1.7 inflowgaugenodelist

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

16.9.1.8 l0_coloutlet

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

16.9.1.9 l0_noutlet

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

16.9.1.10 l0_rowoutlet

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

16.9.1.11 ngauges

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

16.9.1.12 ninflowgauges

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

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

- [mo_mrm_global_variables.f90](#)

16.10 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)

16.10.1 Member Function/Subroutine Documentation

16.10.1.1 bias_dp_1d()

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

16.10.1.2 bias_dp_2d()

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

16.10.1.3 bias_dp_3d()

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

16.10.1.4 bias_sp_1d()

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

16.10.1.5 bias_sp_2d()

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

16.10.1.6 bias_sp_3d()

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

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

- [mo_errormeasures.f90](#)

16.11 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)

16.11.1 Member Function/Subroutine Documentation

16.11.1.1 [central_moment_dp\(\)](#)

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

16.11.1.2 [central_moment_sp\(\)](#)

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

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

- [mo_moment.f90](#)

16.12 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)

16.12.1 Member Function/Subroutine Documentation

16.12.1.1 [central_moment_var_dp\(\)](#)

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

16.12.1.2 central_moment_var_sp()

```
real(sp) function mo_moment::central_moment_var::central_moment_var_sp (
    real(sp), dimension(:), intent(in) x,
    integer(i4), intent(in) n,
    logical, dimension(:), intent(in), optional mask )
```

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

- [mo_moment.f90](#)

16.13 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)

16.13.1 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.

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

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 optional 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	<i>integer, dimension(:) :: classes</i>	classes to categorize data (e.g. months)
in	<i>real(sp/dp), dimension(:) :: data</i>	data to calculate the standard score for
in	<i>logical, dimension(:), optional :: mask</i>	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

16.13.2 Member Function/Subroutine Documentation

16.13.2.1 [classified_standard_score_dp\(\)](#)

```
real(dp) function, dimension(size(data, dim = 1)) mo_standard_score::classified_standard_←
score::classified_standard_score_dp (
    real(dp), dimension(:), intent(in) data,
    integer, dimension(:), intent(in) classes,
    logical, dimension(:), intent(in), optional mask )
```

16.13.2.2 [classified_standard_score_sp\(\)](#)

```
real(sp) function, dimension(size(data, dim = 1)) mo_standard_score::classified_standard_←
score::classified_standard_score_sp (
    real(sp), dimension(:), intent(in) data,
    integer, dimension(:), intent(in) classes,
    logical, dimension(:), intent(in), optional mask )
```

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

- [mo_standard_score.f90](#)

16.14 [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)

16.14.1 Member Function/Subroutine Documentation

16.14.1.1 [corr_dp\(\)](#)

```
real(dp) function, dimension(size(data1)) mo_corr::corr::corr_dp (
    real(dp), dimension(:), intent(in) data1,
```

```
real(dp), dimension(:), intent(in) data2,
integer(i4), intent(out), optional nadjust,
integer(i4), intent(in), optional nhigh,
integer(i4), intent(in), optional nwin )
```

16.14.1.2 corr_sp()

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

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

- [mo_corr.f90](#)

16.15 mo_moment::correlation Interface Reference

Public Member Functions

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

16.15.1 Member Function/Subroutine Documentation

16.15.1.1 correlation_dp()

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

16.15.1.2 correlation_sp()

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

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

- [mo_moment.f90](#)

16.16 mo_moment::covariance Interface Reference

Public Member Functions

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

16.16.1 Member Function/Subroutine Documentation

16.16.1.1 covariance_dp()

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

16.16.1.2 covariance_sp()

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

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

- [mo_moment.f90](#)

16.17 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)

16.17.1 Member Function/Subroutine Documentation

16.17.1.1 crosscoeffk_dp()

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

16.17.1.2 crosscoeffk_sp()

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

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

- [mo_corr.f90](#)

16.18 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)

16.18.1 Member Function/Subroutine Documentation

16.18.1.1 crosscorr_dp()

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

16.18.1.2 crosscorr_sp()

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

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

- [mo_corr.f90](#)

16.19 mo_orderpack::ctrper Interface Reference

Public Member Functions

- subroutine [d_ctrper](#) (XDONT, PCLS)
- subroutine [r_ctrper](#) (XDONT, PCLS)
- subroutine [i_ctrper](#) (XDONT, PCLS)

16.19.1 Member Function/Subroutine Documentation

16.19.1.1 d_ctrper()

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

16.19.1.2 i_ctrper()

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

16.19.1.3 r_ctrper()

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

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

- [mo_orderpack.f90](#)

16.20 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)

16.20.1 Detailed Description

Day-to-month average ([day2mon_average](#))

converts daily time series to monthly

Parameters

in	<i>real(sp/dp) :: daily_data(:)</i>	array of daily time series
in	<i>integer(i4) :: year</i>	year of the starting time
in	<i>integer(i4) :: month</i>	month of the starting time
in	<i>integer(i4) :: day</i>	day of the starting time
in	<i>real(sp/dp) :: mon_average(:)</i>	array of monthly averaged values
in	<i>real(sp/dp) :: misval</i>	missing value definition
in	<i>logical :: rm_misval</i>	switch to exclude missing values

Note

Author

Oldrich Rakovec, Rohini Kumar

Date

Oct 2015

16.20.2 Member Function/Subroutine Documentation

16.20.2.1 day2mon_average_dp()

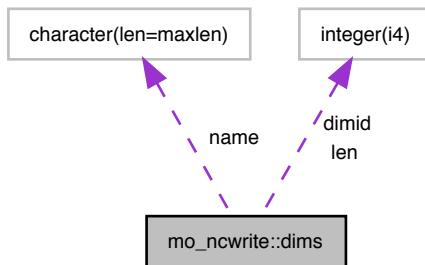
```
subroutine mo_temporal_aggregation::day2mon_average::day2mon_average_dp (
    real(dp), dimension(:), intent(in) daily_data,
    integer(i4), intent(in) years,
    integer(i4), intent(in) months,
    integer(i4), intent(in) days,
    real(dp), dimension(:), intent(inout), allocatable mon_avg,
    real(dp), intent(in), optional misval,
    logical, intent(in), optional rm_misval )
```

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

- [mo_temporal_aggregation.f90](#)

16.21 mo_ncwrite::dims Type Reference

Collaboration diagram for mo_ncwrite::dims:



Public Attributes

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

16.21.1 Member Data Documentation

16.21.1.1 dimid

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

16.21.1.2 len

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

16.21.1.3 name

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

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

- [mo_ncwrite.f90](#)

16.22 mo_ncwrite::dump_ncdf Interface Reference

Public Member Functions

- subroutine [dump_ncdf_1d_sp](#) (filename, arr, append, lfs, netcdf4, deflate_level)
- subroutine [dump_ncdf_2d_sp](#) (filename, arr, append, lfs, netcdf4, deflate_level)
- subroutine [dump_ncdf_3d_sp](#) (filename, arr, append, lfs, netcdf4, deflate_level)
- subroutine [dump_ncdf_4d_sp](#) (filename, arr, append, lfs, netcdf4, deflate_level)
- subroutine [dump_ncdf_5d_sp](#) (filename, arr, append, lfs, netcdf4, deflate_level)
- subroutine [dump_ncdf_1d_dp](#) (filename, arr, append, lfs, netcdf4, deflate_level)
- subroutine [dump_ncdf_2d_dp](#) (filename, arr, append, lfs, netcdf4, deflate_level)
- subroutine [dump_ncdf_3d_dp](#) (filename, arr, append, lfs, netcdf4, deflate_level)
- subroutine [dump_ncdf_4d_dp](#) (filename, arr, append, lfs, netcdf4, deflate_level)
- subroutine [dump_ncdf_5d_dp](#) (filename, arr, append, lfs, netcdf4, deflate_level)
- subroutine [dump_ncdf_1d_i4](#) (filename, arr, append, lfs, netcdf4, deflate_level)
- subroutine [dump_ncdf_2d_i4](#) (filename, arr, append, lfs, netcdf4, deflate_level)
- subroutine [dump_ncdf_3d_i4](#) (filename, arr, append, lfs, netcdf4, deflate_level)
- subroutine [dump_ncdf_4d_i4](#) (filename, arr, append, lfs, netcdf4, deflate_level)
- subroutine [dump_ncdf_5d_i4](#) (filename, arr, append, lfs, netcdf4, deflate_level)

16.22.1 Member Function/Subroutine Documentation

16.22.1.1 dump_netcdf_1d_dp()

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

16.22.1.2 dump_netcdf_1d_i4()

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

16.22.1.3 dump_netcdf_1d_sp()

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

16.22.1.4 dump_netcdf_2d_dp()

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

16.22.1.5 dump_netcdf_2d_i4()

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

16.22.1.6 `dump_netcdf_2d_sp()`

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

16.22.1.7 `dump_netcdf_3d_dp()`

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

16.22.1.8 `dump_netcdf_3d_i4()`

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

16.22.1.9 `dump_netcdf_3d_sp()`

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

16.22.1.10 `dump_netcdf_4d_dp()`

```
subroutine mo_ncwrite::dump_netcdf::dump_netcdf_4d_dp (
    character(len = *), intent(in) filename,
    real(dp), dimension(:, :, :, :, :), intent(in) arr,
    logical, intent(in), optional append,
```

```
logical, intent(in), optional lfs,
logical, intent(in), optional netcdf4,
integer(i4), intent(in), optional deflate_level )
```

16.22.1.11 dump_netcdf_4d_i4()

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

16.22.1.12 dump_netcdf_4d_sp()

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

16.22.1.13 dump_netcdf_5d_dp()

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

16.22.1.14 dump_netcdf_5d_i4()

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

16.22.1.15 dump_netcdf_5d_sp()

```
subroutine mo_ncwrite::dump_netcdf::dump_netcdf_5d_sp (
```

```

character(len = *), intent(in) filename,
real(sp), dimension(:, :, :, :, :), intent(in) arr,
logical, intent(in), optional append,
logical, intent(in), optional lfs,
logical, intent(in), optional netcdf4,
integer(i4), intent(in), optional deflate_level )

```

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

- [mo_ncwrite.f90](#)

16.23 mo_utils::eq Interface Reference

Public Member Functions

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

16.23.1 Member Function/Subroutine Documentation

16.23.1.1 [equal_dp\(\)](#)

```

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

```

16.23.1.2 [equal_sp\(\)](#)

```

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

```

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

- [mo_utils.f90](#)

16.24 mo_utils::equal Interface Reference

Comparison of real values.

Public Member Functions

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

16.24.1 Detailed Description

Comparison of real values.

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

$$\left| \frac{a-b}{b} \right| < \epsilon$$

Parameters

in	<code>real(sp/dp) :: a</code>	First number to compare
in	<code>real(sp/dp) :: b</code>	Second number to compare

Returns

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

Authors

Matthias Cuntz, Juliane Mai

Date

Feb 2014

16.24.2 Member Function/Subroutine Documentation

16.24.2.1 equal_dp()

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

16.24.2.2 equal_sp()

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

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

- [mo_utils.f90](#)

16.25 mo_optimization_utils::eval_interface Interface Reference

Public Member Functions

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

16.25.1 Constructor & Destructor Documentation

16.25.1.1 eval_interface()

```
subroutine mo_optimization_utils::eval_interface::eval_interface (
    real(dp), dimension(:), intent(in) parameterset,
    real(dp), dimension(:, :, ), intent(out), optional, allocatable runoff,
    real(dp), dimension(:, :, ), intent(out), optional, allocatable sm_opti,
    real(dp), dimension(:, :, ), intent(out), optional, allocatable basin_avg_tws,
    real(dp), dimension(:, :, ), intent(out), optional, allocatable neutrons_opti,
    real(dp), dimension(:, :, ), intent(out), optional, allocatable et_opti )
```

References mo_kind::dp.

The documentation for this interface was generated from the following file:

- [mo_optimization_utils.f90](#)

16.26 mo_orderpack::fndnth Interface Reference

Public Member Functions

- real(kind=dp) function [d_fndnth](#) (XDONT, NORD)
- real(kind=sp) function [r_fndnth](#) (XDONT, NORD)
- integer(kind=i4) function [i_fndnth](#) (XDONT, NORD)

16.26.1 Member Function/Subroutine Documentation

16.26.1.1 d_fndnth()

```
real(kind = dp) function mo_orderpack::fndnth::d_fndnth (
    real(kind = dp), dimension (:), intent(in) XDONT,
    integer(kind = i4), intent(in) NORD )
```

16.26.1.2 i_fndnth()

```
integer(kind = i4) function mo_orderpack::fndnth::i_fndnth (
    integer(kind = i4), dimension (:), intent(in) XDONT,
    integer(kind = i4), intent(in) NORD )
```

16.26.1.3 r_fndnth()

```
real(kind = sp) function mo_orderpack::fndnth::r_fndnth (
    real(kind = sp), dimension (:), intent(in) XDONT,
    integer(kind = i4), intent(in) NORD )
```

The documentation for this interface was generated from the following file:

- [mo_orderpack.f90](#)

16.27 mo_corr::four1 Interface Reference

Private Member Functions

- subroutine [four1_sp](#) (data, isign)
- subroutine [four1_dp](#) (data, isign)

16.27.1 Member Function/Subroutine Documentation

16.27.1.1 [four1_dp\(\)](#)

```
subroutine mo_corr::four1::four1_dp (
    complex(dpc), dimension(:), intent(inout) data,
    integer(i4), intent(in) isign )  [private]
```

16.27.1.2 [four1_sp\(\)](#)

```
subroutine mo_corr::four1::four1_sp (
    complex(spc), dimension(:), intent(inout) data,
    integer(i4), intent(in) isign )  [private]
```

The documentation for this interface was generated from the following file:

- [mo_corr.f90](#)

16.28 mo_corr::fourrow Interface Reference

Private Member Functions

- subroutine [fourrow_sp](#) (data, isign)
- subroutine [fourrow_dp](#) (data, isign)

16.28.1 Member Function/Subroutine Documentation

16.28.1.1 [fourrow_dp\(\)](#)

```
subroutine mo_corr::fourrow::fourrow_dp (
    complex(dpc), dimension(:, :), intent(inout) data,
    integer(i4), intent(in) isign )  [private]
```

16.28.1.2 fourrow_sp()

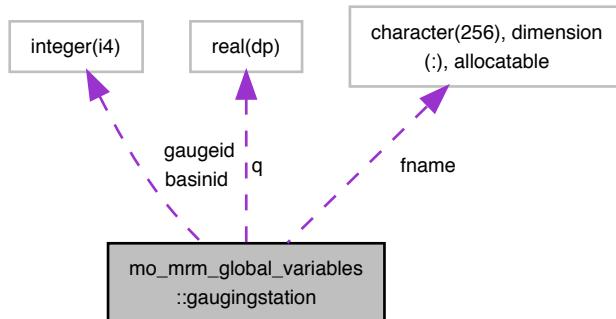
```
subroutine mo_corr::fourrow::fourrow_sp (
    complex(spc), dimension(:, :, ), intent(inout) data,
    integer(i4), intent(in) isign ) [private]
```

The documentation for this interface was generated from the following file:

- [mo_corr.f90](#)

16.29 mo_mrm_global_variables::gaugingstation Type Reference

Collaboration diagram for mo_mrm_global_variables::gaugingstation:



Public Attributes

- integer(i4), dimension(:, allocatable) [basinid](#)
- integer(i4), dimension(:, allocatable) [gaugeid](#)
- character(256), dimension(:, allocatable) [fname](#)
- real(dp), dimension(:, :,), allocatable [q](#)

16.29.1 Member Data Documentation

16.29.1.1 basinid

```
integer(i4), dimension(:, ), allocatable mo_mrm_global_variables::gaugingstation::basinid
```

16.29.1.2 fname

```
character(256), dimension(:, ), allocatable mo_mrm_global_variables::gaugingstation::fname
```

16.29.1.3 gaugeid

```
integer(i4), dimension(:), allocatable mo_mrm_global_variables::gaugingstation::gaugeid
```

16.29.1.4 q

```
real(dp), dimension(:, :), allocatable mo_mrm_global_variables::gaugingstation::q
```

The documentation for this type was generated from the following file:

- [mo_mrm_global_variables.f90](#)

16.30 mo_utils::ge Interface Reference

Public Member Functions

- elemental pure logical function [greaterequal_sp](#) (a, b)
- elemental pure logical function [greaterequal_dp](#) (a, b)

16.30.1 Member Function/Subroutine Documentation

16.30.1.1 greaterequal_dp()

```
elemental pure logical function mo_utils::ge::greaterequal_dp (
    real(dp), intent(in) a,
    real(dp), intent(in) b )
```

16.30.1.2 greaterequal_sp()

```
elemental pure logical function mo_utils::ge::greaterequal_sp (
    real(sp), intent(in) a,
    real(sp), intent(in) b )
```

The documentation for this interface was generated from the following file:

- [mo_utils.f90](#)

16.31 mo_anneal::generate_neighborhood_weight Interface Reference

Private Member Functions

- subroutine [generate_neighborhood_weight_dp](#) (truepara, cum_weight, save_state_xor, iTotCounter, nIT←ERmax, neighborhood)

16.31.1 Member Function/Subroutine Documentation

16.31.1.1 generate_neighborhood_weight_dp()

```
subroutine mo_anneal::generate_neighborhood_weight::generate_neighborhood_weight_dp (
    integer(i4), dimension(:), intent(in) truepara,
    real(dp), dimension(:), intent(in) cum_weight,
    integer(i8), dimension(n_save_state), intent(inout) save_state_xor,
    integer(i4), intent(in) iTotalCounter,
    integer(i4), intent(in) nITERmax,
    logical, dimension(size(cum_weight)), intent(out) neighborhood ) [private]
```

The documentation for this interface was generated from the following file:

- [mo_anneal.f90](#)

16.32 mo_ncread::get_ncvar Interface Reference

Public Member Functions

- subroutine [get_ncvar_0d_sp](#) (Filename, VarName, Dat, fid)
- subroutine [get_ncvar_0d_dp](#) (Filename, VarName, Dat, fid)
- subroutine [get_ncvar_1d_sp](#) (Filename, VarName, Dat, start, a_count, fid)
- subroutine [get_ncvar_1d_dp](#) (Filename, VarName, Dat, start, a_count, fid)
- subroutine [get_ncvar_2d_sp](#) (Filename, VarName, Dat, start, a_count, fid)
- subroutine [get_ncvar_2d_dp](#) (Filename, VarName, Dat, start, a_count, fid)
- subroutine [get_ncvar_3d_sp](#) (Filename, VarName, Dat, start, a_count, fid)
- subroutine [get_ncvar_3d_dp](#) (Filename, VarName, Dat, start, a_count, fid)
- subroutine [get_ncvar_4d_sp](#) (Filename, VarName, Dat, start, a_count, fid)
- subroutine [get_ncvar_4d_dp](#) (Filename, VarName, Dat, start, a_count, fid)
- subroutine [get_ncvar_5d_sp](#) (Filename, VarName, Dat, start, a_count, fid)
- subroutine [get_ncvar_5d_dp](#) (Filename, VarName, Dat, start, a_count, fid)
- subroutine [get_ncvar_0d_i4](#) (Filename, VarName, Dat, fid)
- subroutine [get_ncvar_1d_i4](#) (Filename, VarName, Dat, start, a_count, fid)
- subroutine [get_ncvar_2d_i4](#) (Filename, VarName, Dat, start, a_count, fid)
- subroutine [get_ncvar_3d_i4](#) (Filename, VarName, Dat, start, a_count, fid)
- subroutine [get_ncvar_4d_i4](#) (Filename, VarName, Dat, start, a_count, fid)
- subroutine [get_ncvar_5d_i4](#) (Filename, VarName, Dat, start, a_count, fid)
- subroutine [get_ncvar_0d_i1](#) (Filename, VarName, Dat, fid)
- subroutine [get_ncvar_1d_i1](#) (Filename, VarName, Dat, start, a_count, fid)
- subroutine [get_ncvar_2d_i1](#) (Filename, VarName, Dat, start, a_count, fid)
- subroutine [get_ncvar_3d_i1](#) (Filename, VarName, Dat, start, a_count, fid)
- subroutine [get_ncvar_4d_i1](#) (Filename, VarName, Dat, start, a_count, fid)
- subroutine [get_ncvar_5d_i1](#) (Filename, VarName, Dat, start, a_count, fid)

16.32.1 Member Function/Subroutine Documentation

16.32.1.1 get_ncvar_0d_dp()

```
subroutine mo_ncread::get_ncvar::get_ncvar_0d_dp (
    character(len = *), intent(in) Filename,
    character(len = *), intent(in) VarName,
    real(dp), intent(inout) Dat,
    integer(i4), intent(in), optional fid )
```

16.32.1.2 get_ncvar_0d_i1()

```
subroutine mo_ncread::get_ncvar::get_ncvar_0d_i1 (
    character(len = *), intent(in) Filename,
    character(len = *), intent(in) VarName,
    integer(1), intent(inout) Dat,
    integer(i4), intent(in), optional fid )
```

16.32.1.3 get_ncvar_0d_i4()

```
subroutine mo_ncread::get_ncvar::get_ncvar_0d_i4 (
    character(len = *), intent(in) Filename,
    character(len = *), intent(in) VarName,
    integer(i4), intent(inout) Dat,
    integer(i4), intent(in), optional fid )
```

16.32.1.4 get_ncvar_0d_sp()

```
subroutine mo_ncread::get_ncvar::get_ncvar_0d_sp (
    character(len = *), intent(in) Filename,
    character(len = *), intent(in) VarName,
    real(sp), intent(inout) Dat,
    integer(i4), intent(in), optional fid )
```

16.32.1.5 get_ncvar_1d_dp()

```
subroutine mo_ncread::get_ncvar::get_ncvar_1d_dp (
    character(len = *), intent(in) Filename,
    character(len = *), intent(in) VarName,
    real(dp), dimension(:), intent(inout), allocatable Dat,
    integer(i4), dimension(:), intent(in), optional start,
    integer(i4), dimension(:), intent(in), optional a_count,
    integer(i4), intent(in), optional fid )
```

16.32.1.6 get_ncvar_1d_i1()

```
subroutine mo_ncread::get_ncvar::get_ncvar_1d_i1 (
    character(len = *), intent(in) Filename,
    character(len = *), intent(in) VarName,
    integer(1), dimension(:), intent(inout), allocatable Dat,
    integer(i4), dimension(:), intent(in), optional start,
    integer(i4), dimension(:), intent(in), optional a_count,
    integer(i4), intent(in), optional fid )
```

16.32.1.7 get_ncvar_1d_i4()

```
subroutine mo_ncread::get_ncvar::get_ncvar_1d_i4 (
```

```

character(len = *), intent(in) Filename,
character(len = *), intent(in) VarName,
integer(i4), dimension(:), intent(inout), allocatable Dat,
integer(i4), dimension(:), intent(in), optional start,
integer(i4), dimension(:), intent(in), optional a_count,
integer(i4), intent(in), optional fid )

```

16.32.1.8 `get_ncvar_1d_sp()`

```

subroutine mo_ncread::get_ncvar::get_ncvar_1d_sp (
    character(len = *), intent(in) Filename,
    character(len = *), intent(in) VarName,
    real(sp), dimension(:), intent(inout), allocatable Dat,
    integer(i4), dimension(:), intent(in), optional start,
    integer(i4), dimension(:), intent(in), optional a_count,
    integer(i4), intent(in), optional fid )

```

16.32.1.9 `get_ncvar_2d_dp()`

```

subroutine mo_ncread::get_ncvar::get_ncvar_2d_dp (
    character(len = *), intent(in) Filename,
    character(len = *), intent(in) VarName,
    real(dp), dimension(:, :), intent(inout), allocatable Dat,
    integer(i4), dimension(:, ), intent(in), optional start,
    integer(i4), dimension(:, ), intent(in), optional a_count,
    integer(i4), intent(in), optional fid )

```

16.32.1.10 `get_ncvar_2d_i1()`

```

subroutine mo_ncread::get_ncvar::get_ncvar_2d_i1 (
    character(len = *), intent(in) Filename,
    character(len = *), intent(in) VarName,
    integer(1), dimension(:, :), intent(inout), allocatable Dat,
    integer(i4), dimension(:, ), intent(in), optional start,
    integer(i4), dimension(:, ), intent(in), optional a_count,
    integer(i4), intent(in), optional fid )

```

16.32.1.11 `get_ncvar_2d_i4()`

```

subroutine mo_ncread::get_ncvar::get_ncvar_2d_i4 (
    character(len = *), intent(in) Filename,
    character(len = *), intent(in) VarName,
    integer(i4), dimension(:, :), intent(inout), allocatable Dat,
    integer(i4), dimension(:, ), intent(in), optional start,
    integer(i4), dimension(:, ), intent(in), optional a_count,
    integer(i4), intent(in), optional fid )

```

16.32.1.12 get_ncvar_2d_sp()

```
subroutine mo_ncread::get_ncvar::get_ncvar_2d_sp (
    character(len = *), intent(in) Filename,
    character(len = *), intent(in) VarName,
    real(sp), dimension(:, :, :), intent(inout), allocatable Dat,
    integer(i4), dimension(:, :), intent(in), optional start,
    integer(i4), dimension(:, :), intent(in), optional a_count,
    integer(i4), intent(in), optional fid )
```

16.32.1.13 get_ncvar_3d_dp()

```
subroutine mo_ncread::get_ncvar::get_ncvar_3d_dp (
    character(len = *), intent(in) Filename,
    character(len = *), intent(in) VarName,
    real(dp), dimension(:, :, :, :), intent(inout), allocatable Dat,
    integer(i4), dimension(:, :), intent(in), optional start,
    integer(i4), dimension(:, :), intent(in), optional a_count,
    integer(i4), intent(in), optional fid )
```

16.32.1.14 get_ncvar_3d_i1()

```
subroutine mo_ncread::get_ncvar::get_ncvar_3d_i1 (
    character(len = *), intent(in) Filename,
    character(len = *), intent(in) VarName,
    integer(1), dimension(:, :, :, :), intent(inout), allocatable Dat,
    integer(i4), dimension(:, :), intent(in), optional start,
    integer(i4), dimension(:, :), intent(in), optional a_count,
    integer(i4), intent(in), optional fid )
```

16.32.1.15 get_ncvar_3d_i4()

```
subroutine mo_ncread::get_ncvar::get_ncvar_3d_i4 (
    character(len = *), intent(in) Filename,
    character(len = *), intent(in) VarName,
    integer(i4), dimension(:, :, :, :), intent(inout), allocatable Dat,
    integer(i4), dimension(:, :), intent(in), optional start,
    integer(i4), dimension(:, :), intent(in), optional a_count,
    integer(i4), intent(in), optional fid )
```

16.32.1.16 get_ncvar_3d_sp()

```
subroutine mo_ncread::get_ncvar::get_ncvar_3d_sp (
    character(len = *), intent(in) Filename,
    character(len = *), intent(in) VarName,
    real(sp), dimension(:, :, :, :), intent(inout), allocatable Dat,
    integer(i4), dimension(:, :), intent(in), optional start,
    integer(i4), dimension(:, :), intent(in), optional a_count,
    integer(i4), intent(in), optional fid )
```

16.32.1.17 `get_ncvar_4d_dp()`

```
subroutine mo_ncread::get_ncvar::get_ncvar_4d_dp (
    character(len = *), intent(in) Filename,
    character(len = *), intent(in) VarName,
    real(dp), dimension(:, :, :, :), intent(inout), allocatable Dat,
    integer(i4), dimension(:), intent(in), optional start,
    integer(i4), dimension(:), intent(in), optional a_count,
    integer(i4), intent(in), optional fid )
```

16.32.1.18 `get_ncvar_4d_i1()`

```
subroutine mo_ncread::get_ncvar::get_ncvar_4d_i1 (
    character(len = *), intent(in) Filename,
    character(len = *), intent(in) VarName,
    integer(1), dimension(:, :, :, :), intent(inout), allocatable Dat,
    integer(i4), dimension(:), intent(in), optional start,
    integer(i4), dimension(:), intent(in), optional a_count,
    integer(i4), intent(in), optional fid )
```

16.32.1.19 `get_ncvar_4d_i4()`

```
subroutine mo_ncread::get_ncvar::get_ncvar_4d_i4 (
    character(len = *), intent(in) Filename,
    character(len = *), intent(in) VarName,
    integer(i4), dimension(:, :, :, :), intent(inout), allocatable Dat,
    integer(i4), dimension(:), intent(in), optional start,
    integer(i4), dimension(:), intent(in), optional a_count,
    integer(i4), intent(in), optional fid )
```

16.32.1.20 `get_ncvar_4d_sp()`

```
subroutine mo_ncread::get_ncvar::get_ncvar_4d_sp (
    character(len = *), intent(in) Filename,
    character(len = *), intent(in) VarName,
    real(sp), dimension(:, :, :, :), intent(inout), allocatable Dat,
    integer(i4), dimension(:), intent(in), optional start,
    integer(i4), dimension(:), intent(in), optional a_count,
    integer(i4), intent(in), optional fid )
```

16.32.1.21 `get_ncvar_5d_dp()`

```
subroutine mo_ncread::get_ncvar::get_ncvar_5d_dp (
    character(len = *), intent(in) Filename,
    character(len = *), intent(in) VarName,
    real(dp), dimension(:, :, :, :, :), intent(inout), allocatable Dat,
```

```
integer(i4), dimension(:), intent(in), optional start,
integer(i4), dimension(:), intent(in), optional a_count,
integer(i4), intent(in), optional fid )
```

16.32.1.22 get_ncvar_5d_i1()

```
subroutine mo_ncread::get_ncvar::get_ncvar_5d_i1 (
    character(len = *), intent(in) Filename,
    character(len = *), intent(in) VarName,
    integer(1), dimension(:, :, :, :, :), intent(inout), allocatable Dat,
    integer(i4), dimension(:), intent(in), optional start,
    integer(i4), dimension(:), intent(in), optional a_count,
    integer(i4), intent(in), optional fid )
```

16.32.1.23 get_ncvar_5d_i4()

```
subroutine mo_ncread::get_ncvar::get_ncvar_5d_i4 (
    character(len = *), intent(in) Filename,
    character(len = *), intent(in) VarName,
    integer(i4), dimension(:, :, :, :, :), intent(inout), allocatable Dat,
    integer(i4), dimension(:), intent(in), optional start,
    integer(i4), dimension(:), intent(in), optional a_count,
    integer(i4), intent(in), optional fid )
```

16.32.1.24 get_ncvar_5d_sp()

```
subroutine mo_ncread::get_ncvar::get_ncvar_5d_sp (
    character(len = *), intent(in) Filename,
    character(len = *), intent(in) VarName,
    real(sp), dimension(:, :, :, :, :), intent(inout), allocatable Dat,
    integer(i4), dimension(:), intent(in), optional start,
    integer(i4), dimension(:), intent(in), optional a_count,
    integer(i4), intent(in), optional fid )
```

The documentation for this interface was generated from the following file:

- [mo_ncread.f90](#)

16.33 mo_xor4096::get_timeseed Interface Reference

Public Member Functions

- subroutine [get_timeseed_i4_0d](#) (*seed*)
- subroutine [get_timeseed_i4_1d](#) (*seed*)
- subroutine [get_timeseed_i8_0d](#) (*seed*)
- subroutine [get_timeseed_i8_1d](#) (*seed*)

16.33.1 Member Function/Subroutine Documentation

16.33.1.1 `get_timeseed_i4_0d()`

```
subroutine mo_xor4096::get_timeseed::get_timeseed_i4_0d (
    integer(i4), intent(inout) seed )
```

16.33.1.2 `get_timeseed_i4_1d()`

```
subroutine mo_xor4096::get_timeseed::get_timeseed_i4_1d (
    integer(i4), dimension(:), intent(inout) seed )
```

16.33.1.3 `get_timeseed_i8_0d()`

```
subroutine mo_xor4096::get_timeseed::get_timeseed_i8_0d (
    integer(i8), intent(inout) seed )
```

16.33.1.4 `get_timeseed_i8_1d()`

```
subroutine mo_xor4096::get_timeseed::get_timeseed_i8_1d (
    integer(i8), dimension(:), intent(inout) seed )
```

The documentation for this interface was generated from the following file:

- [mo_xor4096.f90](#)

16.34 `mo_anneal::gettemperature` Interface Reference

GetTemperature.

Public Member Functions

- `real(dp) function gettemperature_dp (paraset, cost, eval, acc_goal, prange, prange_func, samplesize, maskpara, seeds, printflag, weight, maxit, undef_funcval)`

16.34.1 Detailed Description

GetTemperature.

Determines an initial temperature for Simulated Annealing achieving

Parameters

in	<code>REAL(DP), DIMENSION(:) :: paraset</code>	initial (valid) parameter set
in	<code>INTERFACE :: cost_dp</code>	interface calculating the cost function at a given point
in	<code>REAL(DP) :: acc_goal</code>	Acceptance Ratio which has to be achieved
in	<code>REAL(DP), DIMENSION(size(para),2), optional :: prange</code>	lower and upper bound per parameter

Parameters

in	<i>INTERFACE, optional :: prange_func</i>	interface calculating the feasible range for a parameter at a certain point, if ranges are variable
in	<i>INTEGER(I4), optional :: samplesize</i>	number of iterations the estimation of temperature is based on DEFAULT: Max(20_i4*n,250_i4)
in	<i>LOGICAL, DIMENSION(size(para)), optional :: maskpara</i>	maskpara(i) = .true. -> parameter is optimized maskpara(i) = .false. -> parameter is discarded from optimizaion DEFAULT: .true.
in	<i>INTEGER(I4/I8), DIMENSION(2), optional :: seeds</i>	Seeds of random numbers used for random parameter set generation DEFAULT: dependent on current time
in	<i>LOGICAL, optional :: printflag</i>	If .true. detailed command line output is written DEFAULT: .false.
in	<i>REAL(DP), DIMENSION(size(para,1)), optional :: weight</i>	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	<i>LOGICAL, optional :: maxit</i>	minimizing (.false.) or maximizing (.true.) a function DEFAULT: .false. (minimization)
in	<i>REAL(DP), optional :: undef_funcval</i>	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.

16.34.2 Member Function/Subroutine Documentation

16.34.2.1 gettemperature_dp()

```
real(dp) function mo_anneal::gettemperature::gettemperature_dp (
    real(dp), dimension(:), intent(in) paraset,
    procedure(objective_interface), intent(in), pointer cost,
    procedure(eval_interface), intent(in), pointer eval,
    real(dp), intent(in) acc_goal,
    real(dp), dimension(size(paraset, 1), 2), intent(in), optional prange,
    optional prange_func,
```

```

integer(i4), intent(in), optional samplesize,
logical, dimension(size(paraset, 1)), intent(in), optional maskpara,
integer(i8), dimension(2), intent(in), optional seeds,
logical, intent(in), optional printflag,
real(dp), dimension(size(paraset, 1)), intent(in), optional weight,
logical, intent(in), optional maxit,
real(dp), intent(in), optional undef_funcval )

```

The documentation for this interface was generated from the following file:

- [mo_anneal.f90](#)

16.35 mo_utils::greaterequal Interface Reference

Public Member Functions

- elemental pure logical function [greaterequal_sp](#) (a, b)
- elemental pure logical function [greaterequal_dp](#) (a, b)

16.35.1 Member Function/Subroutine Documentation

16.35.1.1 [greaterequal_dp\(\)](#)

```

elemental pure logical function mo_utils::greaterequal::greaterequal_dp (
    real(dp), intent(in) a,
    real(dp), intent(in) b )

```

16.35.1.2 [greaterequal_sp\(\)](#)

```

elemental pure logical function mo_utils::greaterequal::greaterequal_sp (
    real(sp), intent(in) a,
    real(sp), intent(in) b )

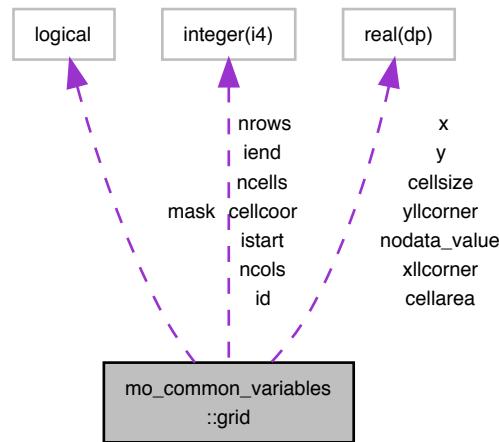
```

The documentation for this interface was generated from the following file:

- [mo_utils.f90](#)

16.36 mo_common_variables::grid Type Reference

Collaboration diagram for mo_common_variables::grid:



Public Attributes

- `integer(i4) ncols`
- `integer(i4) nrows`
- `integer(i4) ncells`
- `real(dp) xllcorner`
- `real(dp) yllcorner`
- `real(dp) cellsize`
- `real(dp) nodata_value`
- `real(dp), dimension(:, :), allocatable x`
- `real(dp), dimension(:, :), allocatable y`
- `logical, dimension(:, :), allocatable mask`
- `integer(i4) istart`
- `integer(i4) iend`
- `integer(i4), dimension(:, :), allocatable cellcoor`
- `real(dp), dimension(:, :), allocatable cellarea`
- `integer(i4), dimension(:, :), allocatable id`

16.36.1 Member Data Documentation

16.36.1.1 cellarea

```
real(dp), dimension(:, :), allocatable mo_common_variables::grid::cellarea
```

16.36.1.2 cellcoor

```
integer(i4), dimension(:, :), allocatable mo_common_variables::grid::cellcoor
```

16.36.1.3 cellsize

```
real(dp) mo_common_variables::grid::cellsize
```

16.36.1.4 id

```
integer(i4), dimension(:, :), allocatable mo_common_variables::grid::id
```

16.36.1.5 iend

```
integer(i4) mo_common_variables::grid::iend
```

16.36.1.6 istart

```
integer(i4) mo_common_variables::grid::istart
```

16.36.1.7 mask

```
logical, dimension(:, :), allocatable mo_common_variables::grid::mask
```

16.36.1.8 ncells

```
integer(i4) mo_common_variables::grid::ncells
```

16.36.1.9 ncols

```
integer(i4) mo_common_variables::grid::ncols
```

16.36.1.10 nodata_value

```
real(dp) mo_common_variables::grid::nodata_value
```

16.36.1.11 nrows

```
integer(i4) mo_common_variables::grid::nrows
```

16.36.1.12 x

```
real(dp), dimension(:, :), allocatable mo_common_variables::grid::x
```

16.36.1.13 xllcorner

```
real(dp) mo_common_variables::grid::xllcorner
```

16.36.1.14 y

```
real(dp), dimension(:, :), allocatable mo_common_variables::grid::y
```

16.36.1.15 yllcorner

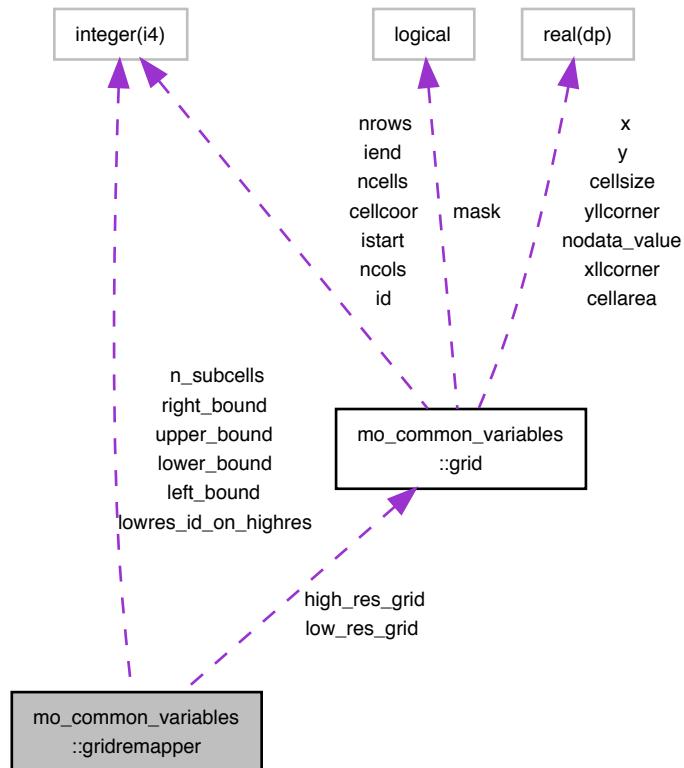
```
real(dp) mo_common_variables::grid::yllcorner
```

The documentation for this type was generated from the following file:

- [mo_common_variables.f90](#)

16.37 mo_common_variables::gridremapper Type Reference

Collaboration diagram for mo_common_variables::gridremapper:



Public Attributes

- type([grid](#)), pointer [high_res_grid](#)
- type([grid](#)), pointer [low_res_grid](#)
- integer(i4), dimension(:), allocatable [lower_bound](#)
- integer(i4), dimension(:), allocatable [upper_bound](#)
- integer(i4), dimension(:), allocatable [left_bound](#)
- integer(i4), dimension(:), allocatable [right_bound](#)
- integer(i4), dimension(:), allocatable [n_subcells](#)
- integer(i4), dimension(:, :), allocatable [lowres_id_on_highres](#)

16.37.1 Member Data Documentation

16.37.1.1 [high_res_grid](#)

```
type(grid), pointer mo_common_variables::gridremapper::high_res_grid
```

16.37.1.2 `left_bound`

```
integer(i4), dimension(:), allocatable mo_common_variables::gridremapper::left_bound
```

16.37.1.3 `low_res_grid`

```
type(grid), pointer mo_common_variables::gridremapper::low_res_grid
```

16.37.1.4 `lower_bound`

```
integer(i4), dimension(:), allocatable mo_common_variables::gridremapper::lower_bound
```

16.37.1.5 `lowres_id_on_highres`

```
integer(i4), dimension(:, :), allocatable mo_common_variables::gridremapper::lowres_id_on_←  
highres
```

16.37.1.6 `n_subcells`

```
integer(i4), dimension(:), allocatable mo_common_variables::gridremapper::n_subcells
```

16.37.1.7 `right_bound`

```
integer(i4), dimension(:), allocatable mo_common_variables::gridremapper::right_bound
```

16.37.1.8 `upper_bound`

```
integer(i4), dimension(:), allocatable mo_common_variables::gridremapper::upper_bound
```

The documentation for this type was generated from the following file:

- [mo_common_variables.f90](#)

16.38 `mo_temporal_aggregation::hour2day_average` Interface Reference

Hour-to-day average ([hour2day_average](#))

Public Member Functions

- subroutine [hour2day_average_dp](#) (hourly_data, yearS, monthS, dayS, hourS, day_avg, misval, rm_misval)

16.38.1 Detailed Description

Hour-to-day average ([hour2day_average](#))

converts hourly time series to daily

Parameters

in	<i>real(sp/dp) :: hourly_data(:)</i>	array of hourly time series
in	<i>integer(i4) :: year</i>	year of the starting time
in	<i>integer(i4) :: month</i>	month of the starting time
in	<i>integer(i4) :: day</i>	day of the starting time
in	<i>integer(i4) :: hour</i>	hour of the starting time
in	<i>real(sp/dp) :: day_average(:)</i>	array of daily averaged values
in	<i>real(sp/dp) :: misval</i>	missing value definition
in	<i>logical :: rm_misval</i>	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

16.38.2 Member Function/Subroutine Documentation

16.38.2.1 [hour2day_average_dp\(\)](#)

```
subroutine mo_temporal_aggregation::hour2day_average::hour2day_average_dp (
    real(dp), dimension(:), intent(in) hourly_data,
    integer(i4), intent(in) years,
    integer(i4), intent(in) months,
    integer(i4), intent(in) days,
    integer(i4), intent(in) hours,
    real(dp), dimension(:), intent(inout), allocatable day_avg,
    real(dp), intent(in), optional misval,
    logical, intent(in), optional rm_misval )
```

The documentation for this interface was generated from the following file:

- [mo_temporal_aggregation.f90](#)

16.39 mo_orderpack::indmed Interface Reference

Public Member Functions

- subroutine [d_indmed](#) (XDONT, INDM)

- subroutine [r_indmed](#) (XDONT, INDM)
- subroutine [i_indmed](#) (XDONT, INDM)

16.39.1 Member Function/Subroutine Documentation

16.39.1.1 [d_indmed\(\)](#)

```
subroutine mo_orderpack::indmed::d_indmed (
    real(kind = dp), dimension (:), intent(in) XDONT,
    integer(kind = i4), intent(out) INDM )
```

16.39.1.2 [i_indmed\(\)](#)

```
subroutine mo_orderpack::indmed::i_indmed (
    integer(kind = i4), dimension (:), intent(in) XDONT,
    integer(kind = i4), intent(out) INDM )
```

16.39.1.3 [r_indmed\(\)](#)

```
subroutine mo_orderpack::indmed::r_indmed (
    real(kind = sp), dimension (:), intent(in) XDONT,
    integer(kind = i4), intent(out) INDM )
```

The documentation for this interface was generated from the following file:

- [mo_orderpack.f90](#)

16.40 mo_orderpack::indnth Interface Reference

Public Member Functions

- integer(kind=i4) function [d_indnth](#) (XDONT, NORD)
- integer(kind=i4) function [r_indnth](#) (XDONT, NORD)
- integer(kind=i4) function [i_indnth](#) (XDONT, NORD)

16.40.1 Member Function/Subroutine Documentation

16.40.1.1 [d_indnth\(\)](#)

```
integer(kind = i4) function mo_orderpack::indnth::d_indnth (
    real(kind = dp), dimension (:), intent(in) XDONT,
    integer(kind = i4), intent(in) NORD )
```

16.40.1.2 i_indnth()

```
integer(kind = i4) function mo_orderpack::indnth::i_indnth (
    integer(kind = i4), dimension (:), intent(in) XDONT,
    integer(kind = i4), intent(in) NORD )
```

16.40.1.3 r_indnth()

```
integer(kind = i4) function mo_orderpack::indnth::r_indnth (
    real(kind = sp), dimension (:), intent(in) XDONT,
    integer(kind = i4), intent(in) NORD )
```

The documentation for this interface was generated from the following file:

- [mo_orderpack.f90](#)

16.41 mo_orderpack::inspar Interface Reference

Public Member Functions

- subroutine [d_inspars](#) (XDONT, NORD)
- subroutine [r_inspars](#) (XDONT, NORD)
- subroutine [i_inspars](#) (XDONT, NORD)

16.41.1 Member Function/Subroutine Documentation

16.41.1.1 d_inspars()

```
subroutine mo_orderpack::inspar::d_inspars (
    real(kind = dp), dimension (:), intent(inout) XDONT,
    integer(kind = i4), intent(in) NORD )
```

16.41.1.2 i_inspars()

```
subroutine mo_orderpack::inspar::i_inspars (
    integer(kind = i4), dimension (:), intent(inout) XDONT,
    integer(kind = i4), intent(in) NORD )
```

16.41.1.3 r_inspars()

```
subroutine mo_orderpack::inspar::r_inspars (
    real(kind = sp), dimension (:), intent(inout) XDONT,
    integer(kind = i4), intent(in) NORD )
```

The documentation for this interface was generated from the following file:

- [mo_orderpack.f90](#)

16.42 mo_orderpack::inssor Interface Reference

Public Member Functions

- subroutine [d_inssor](#) (XDONT)
- subroutine [r_inssor](#) (XDONT)
- subroutine [i_inssor](#) (XDONT)

16.42.1 Member Function/Subroutine Documentation

16.42.1.1 [d_inssor\(\)](#)

```
subroutine mo_orderpack::inssor::d_inssor (
    real(kind = dp), dimension (:), intent(inout) XDONT )
```

16.42.1.2 [i_inssor\(\)](#)

```
subroutine mo_orderpack::inssor::i_inssor (
    integer(kind = i4), dimension (:), intent(inout) XDONT )
```

16.42.1.3 [r_inssor\(\)](#)

```
subroutine mo_orderpack::inssor::r_inssor (
    real(kind = sp), dimension (:), intent(inout) XDONT )
```

The documentation for this interface was generated from the following file:

- [mo_orderpack.f90](#)

16.43 mo_utils::is_finite Interface Reference

.true. if not IEEE Inf, IEEE NaN, nor IEEE Inf nor IEEE NaN, respectively.

Public Member Functions

- elemental pure logical function [is_finite_sp](#) (a)
- elemental pure logical function [is_finite_dp](#) (a)

16.43.1 Detailed Description

.true. if not IEEE Inf, IEEE NaN, nor IEEE Inf nor IEEE NaN, respectively.

Checks for IEEE Inf and IEEE NaN, i.e. Infinity and Not-a-Number.

Wraps to functions of the intrinsic module ieee_arithmetic but gives alternatives for gfortran, which does not provide ieee_arithmetic.

Parameters

in	<i>real(sp/dp) :: x</i>	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

16.43.2 Member Function/Subroutine Documentation

16.43.2.1 is_finite_dp()

```
elemental pure logical function mo_utils::is_finite::is_finite_dp (
    real(dp), intent(in) a )
```

16.43.2.2 is_finite_sp()

```
elemental pure logical function mo_utils::is_finite::is_finite_sp (
    real(sp), intent(in) a )
```

The documentation for this interface was generated from the following file:

- [mo_utils.f90](#)

16.44 mo_utils::is_nan Interface Reference

Public Member Functions

- elemental pure logical function [is_nan_sp \(a\)](#)
- elemental pure logical function [is_nan_dp \(a\)](#)

16.44.1 Member Function/Subroutine Documentation

16.44.1.1 is_nan_dp()

```
elemental pure logical function mo_utils::is_nan::is_nan_dp (
    real(dp), intent(in) a )
```

16.44.1.2 `is_nan_sp()`

```
elemental pure logical function mo_utils::is_nan::is_nan_sp (  
    real(sp), intent(in) a )
```

The documentation for this interface was generated from the following file:

- [mo_utils.f90](#)

16.45 `mo_utils::is_normal` Interface Reference

Public Member Functions

- elemental pure logical function [is_normal_sp \(a\)](#)
- elemental pure logical function [is_normal_dp \(a\)](#)

16.45.1 Member Function/Subroutine Documentation

16.45.1.1 `is_normal_dp()`

```
elemental pure logical function mo_utils::is_normal::is_normal_dp (   
    real(dp), intent(in) a )
```

16.45.1.2 `is_normal_sp()`

```
elemental pure logical function mo_utils::is_normal::is_normal_sp (   
    real(sp), intent(in) a )
```

The documentation for this interface was generated from the following file:

- [mo_utils.f90](#)

16.46 `mo_errormeasures::kge` Interface Reference

Kling-Gupta-Efficiency measure.

Public Member Functions

- `real(dp)` function [kge_dp_1d \(x, y, mask\)](#)
- `real(dp)` function [kge_dp_2d \(x, y, mask\)](#)
- `real(dp)` function [kge_dp_3d \(x, y, mask\)](#)
- `real(sp)` function [kge_sp_1d \(x, y, mask\)](#)
- `real(sp)` function [kge_sp_2d \(x, y, mask\)](#)
- `real(sp)` function [kge_sp_3d \(x, y, mask\)](#)

16.46.1 Detailed Description

Kling-Gupta-Efficiency measure.

The Kling-Gupta model efficiency coefficient KGE is

$$KGE = 1 - \sqrt{((1-r)^2 + (1-\alpha)^2 + (1-\beta)^2)}$$

where

r = Pearson product-moment correlation coefficient

α = ratio of simulated mean to observed mean

β = ratio of simulated standard deviation to observed standard deviation

This three measures are calculated between two arrays (1d, 2d, or 3d). Usually, one is an observation and the second is a modelled variable.

The higher the KGE the better the observation and simulation are matching. The upper limit of KGE is 1.

Therefore, if you apply a minimization algorithm to calibrate regarding KGE you have to use the objective function

$$obj_value = 1.0 - KGE$$

which has then the optimum at 0.0. (Like for the NSE where you always optimize 1-NSE.)

real(sp/dp), dimension(:) :: x, y 1D-array with input numbers
 real(sp/dp), dimension(:, :) :: x, y 2D-array with input numbers
 real(sp/dp), dimension(:, :, :) :: x, y 3D-array with input numbers
 logical :: mask(:) 1D-array of logical values with size(x/y).
 logical :: mask(:, :) 2D-array of logical values with size(x/y).
 logical :: mask(:, :, :) 3D-array of logical values with size(x/y).

Returns

kge — Kling-Gupta-Efficiency (value less equal 1.0)

Note

Input values must be floating points.

Gupta, Hoshin V., et al. "Decomposition of the mean squared error and NSE performance criteria: Implications for improving hydrological modelling." Journal of Hydrology 377.1 (2009): 80-91.

Author

Rohini Kumar

Date

August 2014

16.46.2 Member Function/Subroutine Documentation

16.46.2.1 kge_dp_1d()

```
real(dp) function mo_errormeasures::kge::kge_dp_1d (
    real(dp), dimension(:), intent(in) x,
    real(dp), dimension(:), intent(in) y,
    logical, dimension(:), intent(in), optional mask )
```

16.46.2.2 kge_dp_2d()

```
real(dp) function mo_errormeasures::kge::kge_dp_2d (
    real(dp), dimension(:, :, :), intent(in) x,
    real(dp), dimension(:, :, :), intent(in) y,
    logical, dimension(:, :, :), intent(in), optional mask )
```

16.46.2.3 kge_dp_3d()

```
real(dp) function mo_errormeasures::kge::kge_dp_3d (
    real(dp), dimension(:, :, :, :), intent(in) x,
    real(dp), dimension(:, :, :, :), intent(in) y,
    logical, dimension(:, :, :, :), intent(in), optional mask )
```

16.46.2.4 kge_sp_1d()

```
real(sp) function mo_errormeasures::kge::kge_sp_1d (
    real(sp), dimension(:, :), intent(in) x,
    real(sp), dimension(:, :), intent(in) y,
    logical, dimension(:, :), intent(in), optional mask )
```

16.46.2.5 kge_sp_2d()

```
real(sp) function mo_errormeasures::kge::kge_sp_2d (
    real(sp), dimension(:, :, :), intent(in) x,
    real(sp), dimension(:, :, :), intent(in) y,
    logical, dimension(:, :, :), intent(in), optional mask )
```

16.46.2.6 kge_sp_3d()

```
real(sp) function mo_errormeasures::kge::kge_sp_3d (
    real(sp), dimension(:, :, :, :), intent(in) x,
    real(sp), dimension(:, :, :, :), intent(in) y,
    logical, dimension(:, :, :, :), intent(in), optional mask )
```

The documentation for this interface was generated from the following file:

- [mo_errormeasures.f90](#)

16.47 mo_errormeasures::kgenocorr Interface Reference

Kling-Gupta-Efficiency measure without correlation.

Public Member Functions

- real(dp) function [kgenocorr_dp_1d](#) (x, y, mask)
- real(dp) function [kgenocorr_dp_2d](#) (x, y, mask)

- real(dp) function [kgenocorr_dp_3d](#) (x, y, mask)
- real(sp) function [kgenocorr_sp_1d](#) (x, y, mask)
- real(sp) function [kgenocorr_sp_2d](#) (x, y, mask)
- real(sp) function [kgenocorr_sp_3d](#) (x, y, mask)

16.47.1 Detailed Description

Kling-Gupta-Efficiency measure without correlation.

The modified Kling-Gupta model efficiency coefficient *KGENocorr* is

$$KGENocorr = 1 - \sqrt{((1 - \alpha)^2 + (1 - \beta)^2)}$$

where

α = ratio of simulated mean to observed mean

β = ratio of simulated standard deviation to observed standard deviation

This two measures are calculated between two arrays (1d, 2d, or 3d). Usually, one is an observation and the second is a modelled variable.

The higher the KGENocorr the better the observation and simulation are matching. The upper limit of KGENocorr is 1.

Therefore, if you apply a minimization algorithm to calibrate regarding KGENocorr you have to use the objective function

$$obj_value = 1.0 - KGENocorr$$

which has then the optimum at 0.0. (Like for the NSE where you always optimize 1-NSE.)

real(sp/dp), dimension(:) :: x, y 1D-array with input numbers
 real(sp/dp), dimension(:, :) :: x, y 2D-array with input numbers
 real(sp/dp), dimension(:, :, :) :: x, y 3D-array with input numbers
 logical :: mask(:) 1D-array of logical values with size(x/y).
 logical :: mask(:, :) 2D-array of logical values with size(x/y).
 logical :: mask(:, :, :) 3D-array of logical values with size(x/y).

Returns

kgenocorr — Kling-Gupta-Efficiency without correlation (value less equal 1.0)

Note

Input values must be floating points.

Gupta, Hoshin V., et al. "Decomposition of the mean squared error and NSE performance criteria: Implications for improving hydrological modelling." Journal of Hydrology 377.1 (2009): 80-91.

Author

Rohini Kumar

Date

August 2014

16.47.2 Member Function/Subroutine Documentation

16.47.2.1 kgenocorr_dp_1d()

```
real(dp) function mo_errormeasures::kgenocorr::kgenocorr_dp_1d (
    real(dp), dimension(:), intent(in) x,
    real(dp), dimension(:), intent(in) y,
    logical, dimension(:), intent(in), optional mask )
```

16.47.2.2 kgenocorr_dp_2d()

```
real(dp) function mo_errormeasures::kgenocorr::kgenocorr_dp_2d (
    real(dp), dimension(:, :, :), intent(in) x,
    real(dp), dimension(:, :, :), intent(in) y,
    logical, dimension(:, :, :), intent(in), optional mask )
```

16.47.2.3 kgenocorr_dp_3d()

```
real(dp) function mo_errormeasures::kgenocorr::kgenocorr_dp_3d (
    real(dp), dimension(:, :, :, :), intent(in) x,
    real(dp), dimension(:, :, :, :), intent(in) y,
    logical, dimension(:, :, :, :), intent(in), optional mask )
```

16.47.2.4 kgenocorr_sp_1d()

```
real(sp) function mo_errormeasures::kgenocorr::kgenocorr_sp_1d (
    real(sp), dimension(:, :), intent(in) x,
    real(sp), dimension(:, :), intent(in) y,
    logical, dimension(:, :), intent(in), optional mask )
```

16.47.2.5 kgenocorr_sp_2d()

```
real(sp) function mo_errormeasures::kgenocorr::kgenocorr_sp_2d (
    real(sp), dimension(:, :, :), intent(in) x,
    real(sp), dimension(:, :, :), intent(in) y,
    logical, dimension(:, :, :), intent(in), optional mask )
```

16.47.2.6 kgenocorr_sp_3d()

```
real(sp) function mo_errormeasures::kgenocorr::kgenocorr_sp_3d (
    real(sp), dimension(:, :, :, :), intent(in) x,
    real(sp), dimension(:, :, :, :), intent(in) y,
    logical, dimension(:, :, :, :), intent(in), optional mask )
```

The documentation for this interface was generated from the following file:

- [mo_errormeasures.f90](#)

16.48 mo_moment::kurtosis Interface Reference

Public Member Functions

- real(sp) function [kurtosis_sp](#) (dat, mask)
- real(dp) function [kurtosis_dp](#) (dat, mask)

16.48.1 Member Function/Subroutine Documentation

16.48.1.1 kurtosis_dp()

```
real(dp) function mo_moment::kurtosis::kurtosis_dp (
    real(dp), dimension(:), intent(in) dat,
    logical, dimension(:), intent(in), optional mask )
```

16.48.1.2 kurtosis_sp()

```
real(sp) function mo_moment::kurtosis::kurtosis_sp (
    real(sp), dimension(:), intent(in) dat,
    logical, dimension(:), intent(in), optional mask )
```

The documentation for this interface was generated from the following file:

- [mo_moment.f90](#)

16.49 mo_utils::le Interface Reference

Public Member Functions

- elemental pure logical function [lesserequal_sp](#) (a, b)
- elemental pure logical function [lesserequal_dp](#) (a, b)

16.49.1 Member Function/Subroutine Documentation

16.49.1.1 lesserequal_dp()

```
elemental pure logical function mo_utils::le::lesserequal_dp (
    real(dp), intent(in) a,
    real(dp), intent(in) b )
```

16.49.1.2 lesserequal_sp()

```
elemental pure logical function mo_utils::le::lesserequal_sp (
    real(sp), intent(in) a,
    real(sp), intent(in) b )
```

The documentation for this interface was generated from the following file:

- [mo_utils.f90](#)

16.50 mo_utils::lesserequal Interface Reference

Public Member Functions

- elemental pure logical function [lesserequal_sp](#) (a, b)
- elemental pure logical function [lesserequal_dp](#) (a, b)

16.50.1 Member Function/Subroutine Documentation

16.50.1.1 lesserequal_dp()

```
elemental pure logical function mo_utils::lesserequal::lesserequal_dp (
    real(dp), intent(in) a,
    real(dp), intent(in) b )
```

16.50.1.2 lesserequal_sp()

```
elemental pure logical function mo_utils::lesserequal::lesserequal_sp (
    real(sp), intent(in) a,
    real(sp), intent(in) b )
```

The documentation for this interface was generated from the following file:

- [mo_utils.f90](#)

16.51 mo_linfit::linfit Interface Reference

Fits a straight line to input data by minimizing chi^2.

Public Member Functions

- real(sp) function, dimension(:), allocatable [linfit_sp](#) (x, y, a, b, siga, sigb, chi2, model2)
- real(dp) function, dimension(:), allocatable [linfit_dp](#) (x, y, a, b, siga, sigb, chi2, model2)

16.51.1 Detailed Description

Fits a straight line to input data by minimizing chi^2.

Given a set of data points x(1:ndata), y(1:ndata), fit them to a straight line $y = a + bx$ by minimizing chi2. Model I minimizes y vs. x while Model II takes the geometric mean of y vs. x and x vs. y. Returned is the fitted line at x. Optional returns are a, b and their respective probable uncertainties siga and sigb, and the chi-square chi2.

Parameters

in	<i>real(sp/dp) :: x(:)</i>	1D-array with input x
in	<i>real(sp/dp) :: y(:)</i>	1D-array with input y
in	<i>logical, optional :: model2</i>	If present, use geometric mean regression instead of ordinary least square
out	<i>real(sp/dp), dimension(M) :: a</i>	intercept

Parameters

out	<i>real(sp/dp), dimension(M) :: b</i>	slope
out	<i>real(sp/dp), dimension(M) :: siga</i>	error on intercept
out	<i>real(sp/dp), dimension(M) :: sigb</i>	error on slope
out	<i>real(sp/dp) :: chisq</i>	Minimum chi ²

Returns

real(sp/dp), dimension(:, allocatable :: out — fitted values at *x(:)*.

16.51.2 Member Function/Subroutine Documentation

16.51.2.1 `linfit_dp()`

```
real(dp) function, dimension(:, allocatable mo_linfit::linfit::linfit_dp (
    real(dp), dimension(:, intent(in) x,
    real(dp), dimension(:, intent(in) y,
    real(dp), intent(out), optional a,
    real(dp), intent(out), optional b,
    real(dp), intent(out), optional siga,
    real(dp), intent(out), optional sigb,
    real(dp), intent(out), optional chi2,
    logical, intent(in), optional model2 )
```

16.51.2.2 `linfit_sp()`

```
real(sp) function, dimension(:, allocatable mo_linfit::linfit::linfit_sp (
    real(sp), dimension(:, intent(in) x,
    real(sp), dimension(:, intent(in) y,
    real(sp), intent(out), optional a,
    real(sp), intent(out), optional b,
    real(sp), intent(out), optional siga,
    real(sp), intent(out), optional sigb,
    real(sp), intent(out), optional chi2,
    logical, intent(in), optional model2 )
```

The documentation for this interface was generated from the following file:

- [mo_linfit.f90](#)

16.52 `mo_errormeasures::Innse` Interface Reference

Public Member Functions

- *real(sp)* function [Innse_sp_1d](#) (*x, y, mask*)
- *real(dp)* function [Innse_dp_1d](#) (*x, y, mask*)
- *real(dp)* function [Innse_dp_2d](#) (*x, y, mask*)
- *real(sp)* function [Innse_sp_2d](#) (*x, y, mask*)
- *real(sp)* function [Innse_sp_3d](#) (*x, y, mask*)
- *real(dp)* function [Innse_dp_3d](#) (*x, y, mask*)

16.52.1 Member Function/Subroutine Documentation

16.52.1.1 lnnse_dp_1d()

```
real(dp) function mo_errormeasures::lnnse::lnnse_dp_1d (
    real(dp), dimension(:), intent(in) x,
    real(dp), dimension(:), intent(in) y,
    logical, dimension(:), intent(inout), optional mask )
```

16.52.1.2 lnnse_dp_2d()

```
real(dp) function mo_errormeasures::lnnse::lnnse_dp_2d (
    real(dp), dimension(:, :), intent(in) x,
    real(dp), dimension(:, :), intent(in) y,
    logical, dimension(:, :), intent(inout), optional mask )
```

16.52.1.3 lnnse_dp_3d()

```
real(dp) function mo_errormeasures::lnnse::lnnse_dp_3d (
    real(dp), dimension(:, :, :), intent(in) x,
    real(dp), dimension(:, :, :), intent(in) y,
    logical, dimension(:, :, :), intent(inout), optional mask )
```

16.52.1.4 lnnse_sp_1d()

```
real(sp) function mo_errormeasures::lnnse::lnnse_sp_1d (
    real(sp), dimension(:), intent(in) x,
    real(sp), dimension(:), intent(in) y,
    logical, dimension(:), intent(inout), optional mask )
```

16.52.1.5 lnnse_sp_2d()

```
real(sp) function mo_errormeasures::lnnse::lnnse_sp_2d (
    real(sp), dimension(:, :), intent(in) x,
    real(sp), dimension(:, :), intent(in) y,
    logical, dimension(:, :), intent(inout), optional mask )
```

16.52.1.6 lnnse_sp_3d()

```
real(sp) function mo_errormeasures::lnnse::lnnse_sp_3d (
    real(sp), dimension(:, :, :), intent(in) x,
    real(sp), dimension(:, :, :), intent(in) y,
    logical, dimension(:, :, :), intent(inout), optional mask )
```

The documentation for this interface was generated from the following file:

- `mo_errormeasures.f90`

16.53 `mo_utils::locate` Interface Reference

Find closest values in a monotonic series, returns the indexes.

Public Member Functions

- `integer(i4) function locate_0d_dp (x, y)`
- `integer(i4) function locate_0d_sp (x, y)`
- `integer(i4) function, dimension(:), allocatable locate_1d_dp (x, y)`
- `integer(i4) function, dimension(:), allocatable locate_1d_sp (x, y)`

16.53.1 Detailed Description

Find closest values in a monotonic series, returns the indexes.

Given an array $x(1:n)$, and given a value y , returns a value j such that y is between $x(j)$ and $x(j+1)$. x must be monotonically increasing.
 $j=0$ or $j=N$ is returned to indicate that x is out of range.

Parameters

in	<code>real(dp/sp) :: x(:)</code>	Sorted array
in	<code>real(dp/sp) :: y(:)</code>	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

16.53.2 Member Function/Subroutine Documentation

16.53.2.1 `locate_0d_dp()`

```
integer(i4) function mo_utils::locate::locate_0d_dp (
    real(dp), dimension(:), intent(in) x,
    real(dp), intent(in) y )
```

16.53.2.2 locate_0d_sp()

```
integer(i4) function mo_utils::locate::locate_0d_sp (
    real(sp), dimension(:), intent(in) x,
    real(sp), intent(in) y )
```

16.53.2.3 locate_1d_dp()

```
integer(i4) function, dimension(:), allocatable mo_utils::locate::locate_1d_dp (
    real(dp), dimension(:), intent(in) x,
    real(dp), dimension(:), intent(in) y )
```

16.53.2.4 locate_1d_sp()

```
integer(i4) function, dimension(:), allocatable mo_utils::locate::locate_1d_sp (
    real(sp), dimension(:), intent(in) x,
    real(sp), dimension(:), intent(in) y )
```

The documentation for this interface was generated from the following file:

- [mo_utils.f90](#)

16.54 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)

16.54.1 Member Function/Subroutine Documentation

16.54.1.1 [mae_dp_1d\(\)](#)

```
real(dp) function mo_errormeasures::mae::mae_dp_1d (
    real(dp), dimension(:), intent(in) x,
    real(dp), dimension(:), intent(in) y,
    logical, dimension(:), intent(in), optional mask )
```

16.54.1.2 [mae_dp_2d\(\)](#)

```
real(dp) function mo_errormeasures::mae::mae_dp_2d (
    real(dp), dimension(:, :), intent(in) x,
```

```
real(dp), dimension(:, :, :), intent(in) y,
logical, dimension(:, :, :), intent(in), optional mask )
```

16.54.1.3 mae_dp_3d()

```
real(dp) function mo_errormeasures::mae::mae_dp_3d (
    real(dp), dimension(:, :, :, :), intent(in) x,
    real(dp), dimension(:, :, :, :), intent(in) y,
    logical, dimension(:, :, :, :), intent(in), optional mask )
```

16.54.1.4 mae_sp_1d()

```
real(sp) function mo_errormeasures::mae::mae_sp_1d (
    real(sp), dimension(:, :), intent(in) x,
    real(sp), dimension(:, :), intent(in) y,
    logical, dimension(:, :), intent(in), optional mask )
```

16.54.1.5 mae_sp_2d()

```
real(sp) function mo_errormeasures::mae::mae_sp_2d (
    real(sp), dimension(:, :, :), intent(in) x,
    real(sp), dimension(:, :, :), intent(in) y,
    logical, dimension(:, :, :), intent(in), optional mask )
```

16.54.1.6 mae_sp_3d()

```
real(sp) function mo_errormeasures::mae::mae_sp_3d (
    real(sp), dimension(:, :, :, :), intent(in) x,
    real(sp), dimension(:, :, :, :), intent(in) y,
    logical, dimension(:, :, :, :), intent(in), optional mask )
```

The documentation for this interface was generated from the following file:

- [mo_errormeasures.f90](#)

16.55 mo_mcmc::mcmc Interface Reference

This module is Monte Carlo Markov Chain sampling of a posterior parameter distribution (measurement errors are either known or modeled).

Public Member Functions

- subroutine [mcmc_dp](#) (eval, likelihood, para, rangePar, mcmc_paras, burnin_paras, seed_in, printflag_in, maskpara_in, restart, restart_file, tmp_file, loglike_in, ParaSelectMode_in, iter_burnin_in, iter_mcmc_in, chains_in, stepsize_in)

16.55.1 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 $p_{new_i} = p_{old_i} + \text{Unif}(-\text{stepsize}_i, \text{stepsize}_i)$ (initial : $\text{stepsize}_i = 1.0$ 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 \rightarrow choose one parameter,
 - comput. efficient \rightarrow choose all parameters,
 - moderate accurate & efficient \rightarrow choose half of the parameters
- change parameter(s) based on their stepsize
- decide whether changed parameter set is accepted or rejected:
 - oddsRatio = $\frac{\text{likelihood}(p_{new})}{\text{likelihood}(p_{old})}$
 - random number $r = \text{Uniform}[0,1]$
 - odds Ratio $> 0 \rightarrow$ positive accept
 odds Ratio $> r \rightarrow$ negative accept
 odds Ratio $< r \rightarrow$ reject
- adapt stepsize of parameters changed:

- accepted step: $\text{stepsize_i} = \text{stepsize_i} * \text{acceptance multiplier}$
 - rejected step: $\text{stepsize_i} = \text{stepsize_i} * \text{rejection multiplier}$
 - 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 $< 0.23 \rightarrow \text{acc_mult} = \text{acc_mult} * 0.99$
delete history list
 - acceptance ratio $> 0.44 \rightarrow \text{acc_mult} = \text{acc_mult} * 1.01$
delete history list
 - $0.23 < \text{acceptance ratio} < 0.44$
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 $\sqrt{1/12 * 0.05^2} = \text{Variance of uniform } [\text{acc_ratio } \pm 2.5\%]$)
- 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 \rightarrow choose one parameter,
 - comput. efficient \rightarrow 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:
 - 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	<i>real(dp) :: likelihood(x)</i>	Interface Function which calculates likelihood of given parameter set x
in	<i>real(dp) :: para(:)</i>	Initial parameter set (should be GOOD approximation of best parameter set)
in	<i>real(dp) :: rangePar(size(para),2)</i>	Min/max range of parameters
out	<i>real(dp), allocatable :: mcmc_paras(:, :)</i>	Parameter sets sampled in proper MCMC part of algorithm
out	<i>real(dp), allocatable :: burnin_paras(:, :)</i>	Parameter sets sampled during burn-in part of algorithm
in	<i>integer(i8), optional :: seed_in</i>	User seed to initialise the random number generator (default: none -> initialized with timeseed)
in	<i>logical, optional :: printflag_in</i>	Print of output on command line (default: .False.)
in	<i>logical, optional :: maskpara_in(size(para))</i>	Parameter will be sampled (.True.) or not (.False.) (default: .True.)
in	<i>character(len=*), optional :: tmp_file</i>	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	<i>logical, optional :: loglike_in</i>	true if loglikelihood function is given instead of likelihood function (default: .false.)
in	<i>integer(i4), optional :: ParaSelectMode_in</i>	How many parameters will be changed at once? <ul style="list-style-type: none"> • half of the parameter -> 1_i4 • only one parameter -> 2_i4 • all parameter -> 3_i4 (default: 2_i4)

Parameters

in	<i>integer(i4), optional :: iter_burnin_in</i>	Length of Markov chains of initial burn-in part (default: Max(250, 200*count(maskpara)))
in	<i>integer(i4), optional :: iter_mcmc_in</i>	Length of Markov chains of proper MCMC part (default: 1000 * count(maskpara))
in	<i>integer(i4), optional :: chains_in</i>	number of parallel mcmc chains (default: 5_i4)
in	<i>real(dp), DIMENSION(size(para,1)), optional :: stepsize_in</i>	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.

16.55.2 Member Function/Subroutine Documentation

16.55.2.1 mcmc_dp()

```
subroutine mo_mcmc::mcmc::mcmc_dp (
    procedure(eval\_interface), intent(in), pointer eval,
    procedure(objective\_interface), intent(in), pointer likelihood,
    real(dp), dimension(:), intent(in) para,
    real(dp), dimension(:, :), intent(in) rangePar,
    real(dp), dimension(:, :, :), intent(out), allocatable mcmc_paras,
    real(dp), dimension(:, :, :), intent(out), allocatable burnin_paras,
    integer(i8), intent(in), optional seed_in,
    logical, intent(in), optional printflag_in,
    logical, dimension(size(para, 1)), intent(in), optional maskpara_in,
    logical, intent(in), optional restart,
    character(len = *), intent(in), optional restart_file,
    character(len = *), intent(in), optional tmp_file,
    logical, intent(in), optional loglike_in,
    integer(i4), intent(in), optional ParaSelectMode_in,
    integer(i4), intent(in), optional iter_burnin_in,
    integer(i4), intent(in), optional iter_mcmc_in,
    integer(i4), intent(in), optional chains_in,
    real(dp), dimension(size(para, 1)), intent(in), optional stepsize_in )
```

The documentation for this interface was generated from the following file:

- [mo_mcmc.f90](#)

16.56 mo_mcmc::mcmc_stddev Interface Reference

This module is Monte Carlo Markov Chain sampling of a posterior parameter distribution (measurement errors are neither known nor modeled).

Public Member Functions

- subroutine `mcmc_stddev_dp` (eval, likelihood, para, rangePar, mcmc_paras, burnin_paras, seed_in, printflag_in, maskpara_in, tmp_file, loglike_in, ParaSelectMode_in, iter_burnin_in, iter_mcmc_in, chains_in, stepsize_in)

16.56.1 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 $p_{new_i} = p_{old_i} + \text{Unif}(-\text{stepsize}_i, \text{stepsize}_i)$ (initial : $\text{stepsize}_i = 1.0$ 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 \rightarrow choose one parameter,
 - comput. efficient \rightarrow choose all parameters,
 - moderate accurate & efficient \rightarrow choose half of the parameters
- change parameter(s) based on their stepsize
- decide whether changed parameter set is accepted or rejected:
 - oddsRatio = $\frac{\text{likelihood}(p_{new})}{\text{likelihood}(p_{old})}$
 - random number $r = \text{Uniform}[0,1]$

- odds Ratio > 0 \rightarrow positive accept
 - odds Ratio > r \rightarrow negative accept
 - odds Ratio < r \rightarrow reject
 - adapt stepsize of parameters changed:
 - accepted step: stepsize_i = stepsize_i * acceptance 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 \rightarrow acc_mult = acc_mult * 0.99
delete history list
 - acceptance ratio > 0.44 \rightarrow acc_mult = acc_mult * 1.01
delete history list
 - $0.23 < \text{acceptance ratio} < 0.44$
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 $\sqrt{1/12 * 0.05^2} = \text{Variance of uniform } [\text{acc_ratio } \pm 2.5\%]$)
- 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 \rightarrow choose one parameter,
 - comput. efficient \rightarrow choose all parameters,
 - moderate accurate & efficient \rightarrow choose half of the parameters
2. change parameter(s) based on their stepsize
3. decide whether changed parameter set is accepted or rejected:
 - oddsRatio = $\frac{\text{likelihood}(p_{\text{new}})}{\text{likelihood}(p_{\text{old}})}$
 - random number $r = \text{Uniform}[0,1]$
 - odds Ratio $> 0 \rightarrow$ positive accept
 odds Ratio $> r \rightarrow$ negative accept
 odds Ratio $< r \rightarrow$ reject
4. if step is accepted: save parameter set
5. goto (1)

Parameters

in	<i>real(dp) :: likelihood(x,sigma, stddev_new, likeli_new)</i>	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	<i>real(dp) :: para(:)</i>	Initial parameter set (should be GOOD approximation of best parameter set)
in	<i>real(dp) :: rangePar(size(para),2)</i>	Min/max range of parameters
out	<i>real(dp), allocatable :: mcmc_paras(:, :)</i>	Parameter sets sampled in proper MCMC part of algorithm
out	<i>real(dp), allocatable :: burnin_paras(:, :)</i>	Parameter sets sampled during burn-in part of algorithm
in	<i>integer(i8), optional :: seed_in</i>	User seed to initialise the random number generator (default: none \rightarrow initialized with timeseed)
in	<i>logical, optional :: printflag_in</i>	Print of output on command line (default: .False.)
in	<i>logical, optional :: maskpara_in(size(para))</i>	Parameter will be sampled (.True.) or not (.False.) (default: .True.)
in	<i>character(len=*), optional :: tmp_file</i>	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)
Generated	on June 21, 2018	

Parameters

in	<i>logical, optional :: loglike_in</i>	true if loglikelihood function is given instead of likelihood function (default: .false.)
in	<i>integer(i4), optional :: ParaSelectMode_in</i>	How many parameters will be changed at once? <ul style="list-style-type: none">• half of the parameter -> 1_i4• only one parameter -> 2_i4• all parameter -> 3_i4 (default: 2_i4)
in	<i>integer(i4), optional :: iter_burnin_in</i>	Length of Markov chains of initial burn-in part (default: Max(250, 200*count(maskpara)))
in	<i>integer(i4), optional :: iter_mcmc_in</i>	Length of Markov chains of proper MCMC part (default: 1000 * count(maskpara))
in	<i>integer(i4), optional :: chains_in</i>	number of parallel mcmc chains (default: 5_i4)
in	<i>real(dp), DIMENSION(size(para,1)), optional :: stepsize_in</i>	stepsize for each parameter if given burn-in is discarded (default: none -> adjusted in burn-in)

16.56.2 Member Function/Subroutine Documentation

16.56.2.1 mcmc_stddev_dp()

```
subroutine mo_mcmc::mcmc_stddev::mcmc_stddev_dp (
    procedure(eval_interface), intent(in), pointer eval,
    procedure(objective_interface), intent(in), pointer likelihood,
    real(dp), dimension(:), intent(in) para,
    real(dp), dimension(:, :), intent(in) rangePar,
    real(dp), dimension(:, :, intent(out), allocatable mcmc_paras,
    real(dp), dimension(:, :, intent(out), allocatable burnin_paras,
    integer(i8), intent(in), optional seed_in,
    logical, intent(in), optional printflag_in,
    logical, dimension(size(para, 1)), intent(in), optional maskpara_in,
    character(len = *), intent(in), optional tmp_file,
    logical, intent(in), optional loglike_in,
    integer(i4), intent(in), optional ParaSelectMode_in,
    integer(i4), intent(in), optional iter_burnin_in,
    integer(i4), intent(in), optional iter_mcmc_in,
    integer(i4), intent(in), optional chains_in,
    real(dp), dimension(size(para, 1)), intent(in), optional stepsize_in )
```

The documentation for this interface was generated from the following file:

- [mo_mcmc.f90](#)

16.57 mo_template::mean Interface Reference

The average.

Public Member Functions

- real(sp) function [mean_sp](#) (dat, mask)
- real(dp) function [mean_dp](#) (dat, mask)

16.57.1 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 optional 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

16.57.2 Member Function/Subroutine Documentation

16.57.2.1 mean_dp()

```
real(dp) function mo_template::mean::mean_dp (
    real(dp), dimension(:), intent(in)  dat,
    logical, dimension(:), intent(in), optional mask )
```

16.57.2.2 mean_sp()

```
real(sp) function mo_template::mean::mean_sp (
    real(sp), dimension(:), intent(in)  dat,
    logical, dimension(:), intent(in), optional mask )
```

The documentation for this interface was generated from the following file:

- [mo_template.f90](#)

16.58 mo_moment::mean Interface Reference

Public Member Functions

- real(sp) function [mean_sp](#) (dat, mask)
- real(dp) function [mean_dp](#) (dat, mask)

16.58.1 Member Function/Subroutine Documentation

16.58.1.1 [mean_dp\(\)](#)

```
real(dp) function mo_moment::mean::mean_dp (
    real(dp), dimension(:), intent(in) dat,
    logical, dimension(:), intent(in), optional mask )
```

16.58.1.2 [mean_sp\(\)](#)

```
real(sp) function mo_moment::mean::mean_sp (
    real(sp), dimension(:), intent(in) dat,
    logical, dimension(:), intent(in), optional mask )
```

The documentation for this interface was generated from the following file:

- [mo_moment.f90](#)

16.59 mo_percentile::median Interface Reference

Public Member Functions

- real(sp) function [median_sp](#) (arrin, mask)
- real(dp) function [median_dp](#) (arrin, mask)

16.59.1 Member Function/Subroutine Documentation

16.59.1.1 [median_dp\(\)](#)

```
real(dp) function mo_percentile::median::median_dp (
    real(dp), dimension(:), intent(in) arrin,
    logical, dimension(:), intent(in), optional mask )
```

16.59.1.2 [median_sp\(\)](#)

```
real(sp) function mo_percentile::median::median_sp (
    real(sp), dimension(:), intent(in) arrin,
    logical, dimension(:), intent(in), optional mask )
```

The documentation for this interface was generated from the following file:

- [mo_percentile.f90](#)

16.60 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)

16.60.1 Member Function/Subroutine Documentation

16.60.1.1 [mixed_central_moment_dp\(\)](#)

```
real(dp) function mo_moment::mixed_central_moment::mixed_central_moment_dp ( 
    real(dp), dimension(:), intent(in) x,
    real(dp), dimension(:), intent(in) y,
    integer(i4), intent(in) r,
    integer(i4), intent(in) s,
    logical, dimension(:), intent(in), optional mask )
```

16.60.1.2 [mixed_central_moment_sp\(\)](#)

```
real(sp) function mo_moment::mixed_central_moment::mixed_central_moment_sp ( 
    real(sp), dimension(:), intent(in) x,
    real(sp), dimension(:), intent(in) y,
    integer(i4), intent(in) r,
    integer(i4), intent(in) s,
    logical, dimension(:), intent(in), optional mask )
```

The documentation for this interface was generated from the following file:

- [mo_moment.f90](#)

16.61 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)

16.61.1 Member Function/Subroutine Documentation

16.61.1.1 mixed_central_moment_var_dp()

```
real(dp) function mo_moment::mixed_central_moment_var::mixed_central_moment_var_dp (
    real(dp), dimension(:), intent(in) x,
    real(dp), dimension(:), intent(in) y,
    integer(i4), intent(in) r,
    integer(i4), intent(in) s,
    logical, dimension(:), intent(in), optional mask )
```

16.61.1.2 mixed_central_moment_var_sp()

```
real(sp) function mo_moment::mixed_central_moment_var::mixed_central_moment_var_sp (
    real(sp), dimension(:), intent(in) x,
    real(sp), dimension(:), intent(in) y,
    integer(i4), intent(in) r,
    integer(i4), intent(in) s,
    logical, dimension(:), intent(in), optional mask )
```

The documentation for this interface was generated from the following file:

- [mo_moment.f90](#)

16.62 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)

16.62.1 Member Function/Subroutine Documentation

16.62.1.1 moment_dp()

```
subroutine mo_moment::moment::moment_dp (
    real(dp), dimension(:), intent(in) dat,
    real(dp), intent(out), optional average,
    real(dp), intent(out), optional variance,
    real(dp), intent(out), optional skewness,
    real(dp), intent(out), optional kurtosis,
    real(dp), intent(out), optional mean,
    real(dp), intent(out), optional stddev,
    real(dp), intent(out), optional absdev,
    logical, dimension(:), intent(in), optional mask )
```

16.62.1.2 moment_sp()

```
subroutine mo_moment::moment::moment_sp (
    real(sp), dimension(:), intent(in) dat,
    real(sp), intent(out), optional average,
```

```

real(sp), intent(out), optional variance,
real(sp), intent(out), optional skewness,
real(sp), intent(out), optional kurtosis,
real(sp), intent(out), optional mean,
real(sp), intent(out), optional stddev,
real(sp), intent(out), optional absdev,
logical, dimension(:), intent(in), optional mask )

```

The documentation for this interface was generated from the following file:

- [mo_moment.f90](#)

16.63 mo_orderpack::mrgref Interface Reference

Public Member Functions

- subroutine [d_mrgref](#) (XVALT, IRNGT)
- subroutine [r_mrgref](#) (XVALT, IRNGT)
- subroutine [i_mrgref](#) (XVALT, IRNGT)

16.63.1 Member Function/Subroutine Documentation

16.63.1.1 [d_mrgref\(\)](#)

```

subroutine mo_orderpack::mrgref::d_mrgref (
    real(kind = dp), dimension (:), intent(in) XVALT,
    integer(kind = i4), dimension (:), intent(out) IRNGT )

```

16.63.1.2 [i_mrgref\(\)](#)

```

subroutine mo_orderpack::mrgref::i_mrgref (
    integer(kind = i4), dimension (:), intent(in) XVALT,
    integer(kind = i4), dimension (:), intent(out) IRNGT )

```

16.63.1.3 [r_mrgref\(\)](#)

```

subroutine mo_orderpack::mrgref::r_mrgref (
    real(kind = sp), dimension (:), intent(in) XVALT,
    integer(kind = i4), dimension (:), intent(out) IRNGT )

```

The documentation for this interface was generated from the following file:

- [mo_orderpack.f90](#)

16.64 mo_orderpack::mrgrnk Interface Reference

Public Member Functions

- subroutine [d_mrgrnk](#) (XDONT, IRNGT)
- subroutine [r_mrgrnk](#) (XDONT, IRNGT)
- subroutine [i_mrgrnk](#) (XDONT, IRNGT)

16.64.1 Member Function/Subroutine Documentation

16.64.1.1 [d_mrgrnk\(\)](#)

```
subroutine mo_orderpack::mrgrnk::d_mrgrnk (
    real(kind = dp), dimension (:), intent(in) XDONT,
    integer(kind = i4), dimension (:), intent(out) IRNGT )
```

16.64.1.2 [i_mrgrnk\(\)](#)

```
subroutine mo_orderpack::mrgrnk::i_mrgrnk (
    integer(kind = i4), dimension (:), intent(in) XDONT,
    integer(kind = i4), dimension (:), intent(out) IRNGT )
```

16.64.1.3 [r_mrgrnk\(\)](#)

```
subroutine mo_orderpack::mrgrnk::r_mrgrnk (
    real(kind = sp), dimension (:), intent(in) XDONT,
    integer(kind = i4), dimension (:), intent(out) IRNGT )
```

The documentation for this interface was generated from the following file:

- [mo_orderpack.f90](#)

16.65 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)

16.65.1 Member Function/Subroutine Documentation

16.65.1.1 mse_dp_1d()

```
real(dp) function mo_errormeasures::mse::mse_dp_1d (
    real(dp), dimension(:), intent(in) x,
    real(dp), dimension(:), intent(in) y,
    logical, dimension(:), intent(in), optional mask )
```

16.65.1.2 mse_dp_2d()

```
real(dp) function mo_errormeasures::mse::mse_dp_2d (
    real(dp), dimension(:, :), intent(in) x,
    real(dp), dimension(:, :), intent(in) y,
    logical, dimension(:, :, intent(in), optional mask )
```

16.65.1.3 mse_dp_3d()

```
real(dp) function mo_errormeasures::mse::mse_dp_3d (
    real(dp), dimension(:, :, :), intent(in) x,
    real(dp), dimension(:, :, :), intent(in) y,
    logical, dimension(:, :, :, intent(in), optional mask )
```

16.65.1.4 mse_sp_1d()

```
real(sp) function mo_errormeasures::mse::mse_sp_1d (
    real(sp), dimension(:), intent(in) x,
    real(sp), dimension(:), intent(in) y,
    logical, dimension(:), intent(in), optional mask )
```

16.65.1.5 mse_sp_2d()

```
real(sp) function mo_errormeasures::mse::mse_sp_2d (
    real(sp), dimension(:, :), intent(in) x,
    real(sp), dimension(:, :), intent(in) y,
    logical, dimension(:, :, intent(in), optional mask )
```

16.65.1.6 mse_sp_3d()

```
real(sp) function mo_errormeasures::mse::mse_sp_3d (
    real(sp), dimension(:, :, :), intent(in) x,
    real(sp), dimension(:, :, :), intent(in) y,
    logical, dimension(:, :, :, intent(in), optional mask )
```

The documentation for this interface was generated from the following file:

- [mo_errormeasures.f90](#)

16.66 mo_orderpack::mulcnt Interface Reference

Public Member Functions

- subroutine [d_mulcnt](#) (XDONT, IMULT)
- subroutine [r_mulcnt](#) (XDONT, IMULT)
- subroutine [i_mulcnt](#) (XDONT, IMULT)

16.66.1 Member Function/Subroutine Documentation

16.66.1.1 [d_mulcnt\(\)](#)

```
subroutine mo_orderpack::mulcnt::d_mulcnt (
    real(kind = dp), dimension (:), intent(in) XDONT,
    integer(kind = i4), dimension (:), intent(out) IMULT )
```

16.66.1.2 [i_mulcnt\(\)](#)

```
subroutine mo_orderpack::mulcnt::i_mulcnt (
    integer(kind = i4), dimension (:), intent(in) XDONT,
    integer(kind = i4), dimension (:), intent(out) IMULT )
```

16.66.1.3 [r_mulcnt\(\)](#)

```
subroutine mo_orderpack::mulcnt::r_mulcnt (
    real(kind = sp), dimension (:), intent(in) XDONT,
    integer(kind = i4), dimension (:), intent(out) IMULT )
```

The documentation for this interface was generated from the following file:

- [mo_orderpack.f90](#)

16.67 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)

16.67.1 Member Function/Subroutine Documentation

16.67.1.1 n_element_dp()

```
real(dp) function mo_percentile::n_element::n_element_dp (
    real(dp), dimension(:), intent(in) idat,
    integer(i4), intent(in) n,
    logical, dimension(:), intent(in), optional mask,
    real(dp), intent(out), optional before,
    real(dp), intent(out), optional after,
    real(dp), intent(out), optional previous,
    real(dp), intent(out), optional next )
```

16.67.1.2 n_element_sp()

```
real(sp) function mo_percentile::n_element::n_element_sp (
    real(sp), dimension(:), intent(in) idat,
    integer(i4), intent(in) n,
    logical, dimension(:), intent(in), optional mask,
    real(sp), intent(out), optional before,
    real(sp), intent(out), optional after,
    real(sp), intent(out), optional previous,
    real(sp), intent(out), optional next )
```

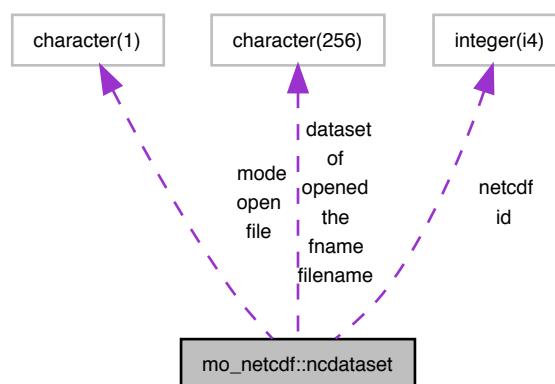
The documentation for this interface was generated from the following file:

- [mo_percentile.f90](#)

16.68 mo_ncdf::ncdataset Interface Reference

Provides basic file modification functionality.

Collaboration diagram for mo_ncdf::ncdataset:



Public Member Functions

- procedure, public `initncdataset`
- procedure, public `getnvariables`
- 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 dataset.
- procedure, public `setdimension`

Create a new dimension.
- generic, public `setattribute` => `setGlobalAttributeChar`, `setGlobalAttributeI8`, `setGlobalAttributeI16`, `setGlobalAttributeI32`, `setGlobalAttributeI64`, `setGlobalAttributeF32`, `setGlobalAttributeF64`

Create a new global Attribute.
- generic, public `getattribute` => `getGlobalAttributeChar`, `getGlobalAttributeI8`, `getGlobalAttributeI16`, `getGlobalAttributeI32`, `getGlobalAttributeI64`, `getGlobalAttributeF32`, `getGlobalAttributeF64`

Retrieve global attribute value.
- generic, public `getdimension` => `getDimensionById`, `getDimensionByName`

Retrieve NcDimension.
- generic, public `setvariable` => `setVariableWithNames`, `setVariableWithTypes`, `setVariableWithIds`

Create a NetCDF variable.
- generic, public `getvariable` => `getVariableByName`

Retrieve NcVariable.

Public Attributes

- character(256) `fname`
- character(256) `filename`
- character(256) `of`
- character(256) `the`
- character(256) `opened`
- character(256) `dataset`
- character(1) `mode`
- character(1) `file`
- character(1) `open`
- integer(i4) `id`
- integer(i4) `netcdf`

Private Member Functions

- procedure, private `setglobalattributechar`
- procedure, private `setglobalattributei8`
- procedure, private `setglobalattributei16`
- procedure, private `setglobalattributei32`
- procedure, private `setglobalattributei64`

- procedure, private `setglobalattribute32`
- procedure, private `setglobalattribute64`
- 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`

16.68.1 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	<code>character(*) :: fname</code>	
in	<code>character(1) :: mode</code>	

Returns

`"type(NcDataset)"`

16.68.2 Member Function/Subroutine Documentation

16.68.2.1 close()

```
procedure, public mo_ncdf::ncdataset::close ( )
```

Close the dataset.

Close the NetCDF dataset. The program will terminate abruptly if the file cannot be closed correctly.

Author

David Schaefer

Date

June 2015

16.68.2.2 getattribute()

```
generic, public mo_netcdf::ncdataset::getattribute ( )
```

Retrieve global attribute value.

Retrieve the value for a global attribute specified by its name. The program will terminate abruptly if the attribute does not exist.

Parameters

in	<i>character(*) :: name</i>	
out	<i>character(*)/integer(i4)/real(sp)/real(dp) :: value</i>	

Author

David Schaefer

Date

June 2015

16.68.2.3 getdimension()

```
generic, public mo_netcdf::ncdataset::getdimension ( )
```

Retrieve NcDimension.

Retrieve the NcDimension derived type for the dimension specified by its name or id. The program will terminate abruptly if no such dimension exists.

Parameters

in	<i>character(*)/integer(i4) :: name/id</i>	
----	--	--

Returns

NcDimension

Author

David Schaefer

Date

June 2015

16.68.2.4 getdimensionbyid()

```
procedure, private mo_netcdf::ncdataset::getdimensionbyid ( ) [private]
```

16.68.2.5 getdimensionbyname()

```
procedure, private mo_ncdf::ncdataset::getdimensionbyname ( ) [private]
```

16.68.2.6 getglobalattributechar()

```
procedure, private mo_ncdf::ncdataset::getglobalattributechar ( ) [private]
```

16.68.2.7 getglobalattribute32()

```
procedure, private mo_ncdf::ncdataset::getglobalattribute32 ( ) [private]
```

16.68.2.8 getglobalattribute64()

```
procedure, private mo_ncdf::ncdataset::getglobalattribute64 ( ) [private]
```

16.68.2.9 getglobalattributei16()

```
procedure, private mo_ncdf::ncdataset::getglobalattributei16 ( ) [private]
```

16.68.2.10 getglobalattributei32()

```
procedure, private mo_ncdf::ncdataset::getglobalattributei32 ( ) [private]
```

16.68.2.11 getglobalattributei64()

```
procedure, private mo_ncdf::ncdataset::getglobalattributei64 ( ) [private]
```

16.68.2.12 getglobalattributei8()

```
procedure, private mo_ncdf::ncdataset::getglobalattributei8 ( ) [private]
```

16.68.2.13 getnovariables()

```
procedure, public mo_ncdf::ncdataset::getnovariables ( )
```

16.68.2.14 getunlimiteddimension()

```
procedure, public mo_netcdf::ncdataset::getunlimiteddimension ( )
```

Return the unlimited dimension of the dataset.

Returns the NcDimension derived type of the unlimited dimension. The program will terminate abruptly if no such dimension exists.

Parameters

in	<i>character(*) :: name</i>	<input type="button" value=""/>
----	-----------------------------	---------------------------------

Returns

"logical"

Author

David Schaefer

Date

June 2015

16.68.2.15 getvariable()

```
generic, public mo_netcdf::ncdataset::getvariable ( )
```

Retrieve NcVariable.

Retrieve the NcVariable derived type for the variable specified by its name. The program will terminate abruptly if no such dimension exists.

Parameters

in	<i>character(*) :: name</i>	<input type="button" value=""/>
----	-----------------------------	---------------------------------

Returns

NcVariable

Author

David Schaefer

Date

June 2015

16.68.2.16 getvariablebyname()

```
procedure, private mo_netcdf::ncdataset::getvariablebyname ( ) [private]
```

16.68.2.17 getvariableids()

```
procedure, public mo_ncdf::ncdataset::getvariableids ( )
```

16.68.2.18 getvariables()

```
procedure, public mo_ncdf::ncdataset::getvariables ( )
```

16.68.2.19 hasdimension()

```
procedure, public mo_ncdf::ncdataset::hasdimension ( )
```

Check if dimension exists.

Returns true if a dimension with the given name exists, false otherwise.

Parameters

in	<i>character(*) :: name</i>	<input type="button" value=""/>
----	-----------------------------	---------------------------------

Returns

"logical"

Author

David Schaefer

Date

June 2015

16.68.2.20 hasvariable()

```
procedure, public mo_ncdf::ncdataset::hasvariable ( )
```

Check if variable exists.

Returns true if a variable with the given name exists, false otherwise.

Parameters

in	<i>character(*) :: name</i>	<input type="button" value=""/>
----	-----------------------------	---------------------------------

Returns

"logical"

Author

David Schaefer

Date

June 2015

16.68.2.21 initncdataset()

```
procedure, public mo_netcdf::ncdataset::initncdataset ( )
```

16.68.2.22 isunlimited()

```
procedure, public mo_netcdf::ncdataset::isunlimited ( )
```

Check if the dataset is unlimited.

Returns true if the dataset contains an unlimited dimension, false otherwise.

Returns

"logical"

Author

David Schaefer

Date

June 2015

16.68.2.23 setattribute()

```
generic, public mo_netcdf::ncdataset::setattribute ( )
```

Create a new global Attribute.

Create a new global attribute from given name and value. The program will terminate abruptly if the attribute cannot be created.

Parameters

in	<i>character(*) :: name</i>	
in	<i>character(*)/integer(i4)/real(sp)/real(dp) :: value</i>	

Author

David Schaefer

Date

June 2015

16.68.2.24 setdimension()

```
procedure, public mo_ncdf::ncdataset::setdimension ( )
```

Create a new dimension.

Create a new dimension of given length. A length < 0 indicates an unlimited dimension. The program will terminate abruptly if the dimension cannot be created.

Parameters

in	<i>character(*) :: name</i>	
in	<i>integer(i4) :: length</i>	

Returns

NcDimension

Author

David Schaefer

Date

June 2015

16.68.2.25 setglobalattributechar()

```
procedure, private mo_ncdf::ncdataset::setglobalattributechar ( ) [private]
```

16.68.2.26 setglobalattribute32()

```
procedure, private mo_ncdf::ncdataset::setglobalattribute32 ( ) [private]
```

16.68.2.27 setglobalattribute64()

```
procedure, private mo_ncdf::ncdataset::setglobalattribute64 ( ) [private]
```

16.68.2.28 setglobalattributei16()

```
procedure, private mo_ncdf::ncdataset::setglobalattributei16 ( ) [private]
```

16.68.2.29 setglobalattributei32()

```
procedure, private mo_netcdf::ncdataset::setglobalattributei32 ( ) [private]
```

16.68.2.30 setglobalattributei64()

```
procedure, private mo_netcdf::ncdataset::setglobalattributei64 ( ) [private]
```

16.68.2.31 setglobalattributei8()

```
procedure, private mo_netcdf::ncdataset::setglobalattributei8 ( ) [private]
```

16.68.2.32 setvariable()

```
generic, public mo_netcdf::ncdataset::setvariable ( )
```

Create a NetCDF variable.

Create a NetCDF Variable with given name, data type and dimensions. All optional arguments to the nf90_def_var function are supported. The program will terminate 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	<i>character(*) :: name</i>	
in	<i>character(3) :: dtype</i>	
in	<i>integer(i4)/character(*)/type(NcDataset) :: dimensions(:)</i>	
in	<i>logical :: contiguous</i>	
in	<i>integer(i4) :: chunksizes(:)</i>	
in	<i>integer(i4) :: deflate_level</i>	
in	<i>logical :: shuffle</i>	
in	<i>logical :: fletcher32</i>	
in	<i>integer(i4) :: endianess</i>	
in	<i>integer(i4) :: cache_size</i>	
in	<i>integer(i4) :: cache_nelems</i>	
in	<i>integer(i4) :: cache_preemption</i>	

Returns

NcVariable

Author

David Schaefer

Date

June 2015

16.68.2.33 setvariablewithids()

```
procedure, private mo_ncdf::ncdataset::setvariablewithids ( ) [private]
```

16.68.2.34 setvariablewithnames()

```
procedure, private mo_ncdf::ncdataset::setvariablewithnames ( ) [private]
```

16.68.2.35 setvariablewithtypes()

```
procedure, private mo_ncdf::ncdataset::setvariablewithtypes ( ) [private]
```

16.68.3 Member Data Documentation**16.68.3.1 dataset**

```
character(256) mo_ncdf::ncdataset::dataset
```

16.68.3.2 file

```
character(1) mo_ncdf::ncdataset::file
```

16.68.3.3 filename

```
character(256) mo_ncdf::ncdataset::filename
```

16.68.3.4 fname

```
character(256) mo_ncdf::ncdataset::fname
```

16.68.3.5 id

```
integer(i4) mo_ncdf::ncdataset::id
```

16.68.3.6 mode

```
character(1) mo_netcdf::ncdataset::mode
```

16.68.3.7 netcdf

```
integer(i4) mo_netcdf::ncdataset::netcdf
```

16.68.3.8 of

```
character(256) mo_netcdf::ncdataset::of
```

16.68.3.9 open

```
character(1) mo_netcdf::ncdataset::open
```

16.68.3.10 opened

```
character(256) mo_netcdf::ncdataset::opened
```

16.68.3.11 the

```
character(256) mo_netcdf::ncdataset::the
```

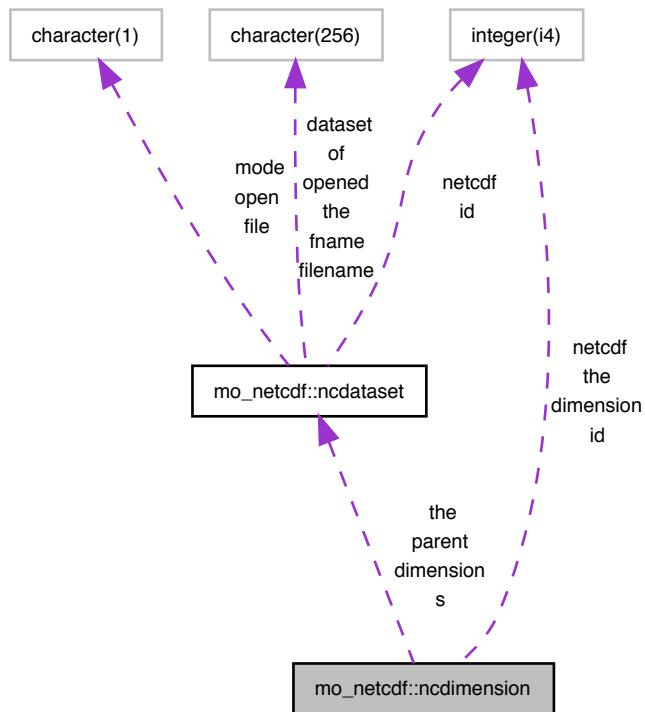
The documentation for this interface was generated from the following file:

- [mo_netcdf.f90](#)

16.69 mo_netcdf::ncdimension Type Reference

Provides the dimension access functionality.

Collaboration diagram for mo_netcdf::ncdimension:



Public Member Functions

- procedure, public **initncvariable**
 - procedure, public **getname** => getVariableName
 - procedure, public **getnodimensions**

Retrieve the number of dimensions.
 - procedure, public **getdimensions** => getVariableDimensions

Retrieve the variable dimensions.
 - procedure, public **getshape** => getVariableShape

Retrieve the shape of the variable.
 - procedure, public **getdtype** => getVariableDtype

Retrieve the variable data type.
 - procedure, public **hasattribute**

Check if attribute exists.
 - procedure, public **isunlimited** => isUnlimitedVariable

Check if the variable is unlimited.
 - generic, public **setdata** => setDataScalarI8, setData1dI8, setData2dI8, setData3dI8, setData4dI8, setData5dI8, setDataScalarI16, setData1dI16, setData2dI16, setData3dI16, setData4dI16, setData5dI16, setDataScalarI32, setData1dI32, setData2dI32, setData3dI32, setData4dI32, setData5dI32, setDataScalarI64, setData1dI64, setData2dI64, setData3dI64, setData4dI64, setData5dI64, setDataScalarF32, setData1dF32, setData2dF32, setData3dF32, setData4dF32, setData5dF32, setDataScalarF64, setData1dF64, setData2dF64, setData3dF64, setData4dF64, setData5dF64

Write data to variable.

- generic, public `getdata` => `getDataScalarI8`, `getData1di8`, `getData2di8`, `getData3di8`, `getData4di8`, `getData5di8`, `getDataScalarI16`, `getData1di16`, `getData2di16`, `getData3di16`, `getData4di16`, `getData5di16`, `getDataScalarI32`, `getData1di32`, `getData2di32`, `getData3di32`, `getData4di32`, `getData5di32`, `getDataScalarI64`, `getData1di64`, `getData2di64`, `getData3di64`, `getData4di64`, `getData5di64`, `getDataScalarF32`, `getData1dF32`, `getData2dF32`, `getData3dF32`, `getData4dF32`, `getData5dF32`, `getDataScalarF64`, `getData1dF64`, `getData2dF64`, `getData3dF64`, `getData4dF64`, `getData5dF64`
 - Retrieve data.*
- generic, public `setfillvalue` => `setVariableFillValueI8`, `setVariableFillValueI16`, `setVariableFillValueI32`, `setVariableFillValueI64`, `setVariableFillValueF32`, `setVariableFillValueF64`
 - Set the variable fill value.*
- generic, public `getfillvalue` => `getVariableFillValueI8`, `getVariableFillValueI16`, `getVariableFillValueI32`, `getVariableFillValueI64`, `getVariableFillValueF32`, `getVariableFillValueF64`
 - Retrieve the variable fill value.*
- generic, public `setattribute` => `setVariableAttributeChar`, `setVariableAttributeI8`, `setVariableAttributeI16`, `setVariableAttributeI32`, `setVariableAttributeI64`, `setVariableAttributeF32`, `setVariableAttributeF64`
 - Create a new variable attribute.*
- generic, public `getattribute` => `getVariableAttributeChar`, `getVariableAttributeI8`, `getVariableAttributeI16`, `getVariableAttributeI32`, `getVariableAttributeI64`, `getVariableAttributeF32`, `getVariableAttributeF64`
 - Retrieve variable attribute value.*

Public Attributes

- integer(i4) `id`
- integer(i4) `the`
- integer(i4) `netcdf`
- integer(i4) `dimension`
- type(`ncdataset`) `parent`
- type(`ncdataset`) `the`
- type(`ncdataset`) `dimension`
- type(`ncdataset`) `s`

Private Member Functions

- procedure, private `setvariableattributechar`
- procedure, private `setvariableattributei8`
- procedure, private `setvariableattributei16`
- procedure, private `setvariableattributei32`
- procedure, private `setvariableattributei64`
- procedure, private `setvariableattributef32`
- procedure, private `setvariableattributef64`
- procedure, private `getvariableattributechar`
- procedure, private `getvariableattributei8`
- procedure, private `getvariableattributei16`
- procedure, private `getvariableattributei32`
- procedure, private `getvariableattributei64`
- procedure, private `getvariableattributef32`
- procedure, private `getvariableattributef64`
- procedure, private `setdatascalarI8`
- procedure, private `setdata1di8`
- procedure, private `setdata2di8`
- procedure, private `setdata3di8`
- procedure, private `setdata4di8`
- procedure, private `setdata5di8`

- procedure, private `setdatascalari16`
- procedure, private `setdata1di16`
- procedure, private `setdata2di16`
- procedure, private `setdata3di16`
- procedure, private `setdata4di16`
- procedure, private `setdata5di16`
- procedure, private `setdatascalari32`
- procedure, private `setdata1di32`
- procedure, private `setdata2di32`
- procedure, private `setdata3di32`
- procedure, private `setdata4di32`
- procedure, private `setdata5di32`
- procedure, private `setdatascalari64`
- procedure, private `setdata1di64`
- procedure, private `setdata2di64`
- procedure, private `setdata3di64`
- procedure, private `setdata4di64`
- procedure, private `setdata5di64`
- procedure, private `setdatascalarf32`
- procedure, private `setdata1df32`
- procedure, private `setdata2df32`
- procedure, private `setdata3df32`
- procedure, private `setdata4df32`
- procedure, private `setdata5df32`
- procedure, private `setdatascalarf64`
- procedure, private `setdata1df64`
- procedure, private `setdata2df64`
- procedure, private `setdata3df64`
- procedure, private `setdata4df64`
- procedure, private `setdata5df64`
- procedure, private `getdatascalari8`
- procedure, private `getdata1di8`
- procedure, private `getdata2di8`
- procedure, private `getdata3di8`
- procedure, private `getdata4di8`
- procedure, private `getdata5di8`
- procedure, private `getdatascalari16`
- procedure, private `getdata1di16`
- procedure, private `getdata2di16`
- procedure, private `getdata3di16`
- procedure, private `getdata4di16`
- procedure, private `getdata5di16`
- procedure, private `getdatascalari32`
- procedure, private `getdata1di32`
- procedure, private `getdata2di32`
- procedure, private `getdata3di32`
- procedure, private `getdata4di32`
- procedure, private `getdata5di32`
- procedure, private `getdatascalari64`
- procedure, private `getdata1di64`
- procedure, private `getdata2di64`
- procedure, private `getdata3di64`
- procedure, private `getdata4di64`
- procedure, private `getdata5di64`
- procedure, private `getdatascalarf32`

- procedure, private `getdata1df32`
- procedure, private `getdata2df32`
- procedure, private `getdata3df32`
- procedure, private `getdata4df32`
- procedure, private `getdata5df32`
- procedure, private `getdatascalarf64`
- procedure, private `getdata1df64`
- procedure, private `getdata2df64`
- procedure, private `getdata3df64`
- procedure, private `getdata4df64`
- procedure, private `getdata5df64`
- procedure, private `setvariablefillvaluei8`
- procedure, private `setvariablefillvaluei16`
- procedure, private `setvariablefillvaluei32`
- procedure, private `setvariablefillvaluei64`
- procedure, private `setvariablefillvaluef32`
- procedure, private `setvariablefillvaluef64`
- procedure, private `getvariablefillvaluei8`
- procedure, private `getvariablefillvaluei16`
- procedure, private `getvariablefillvaluei32`
- procedure, private `getvariablefillvaluei64`
- procedure, private `getvariablefillvaluef32`
- procedure, private `getvariablefillvaluef64`

16.69.1 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".

16.69.2 Member Function/Subroutine Documentation

16.69.2.1 `getattribute()`

```
generic, public mo_netcdf::ncdimension::getattribute ( )
```

Retrieve variable attribute value.

Retrieve the value for a variable attribute specified by its name. The program will terminate abruptly if the attribute does not exist.

Parameters

<code>in</code>	<code>character(*) :: name</code>	
<code>out</code>	<code>character(*)/integer(i4)/real(sp)/real(dp) :: value</code>	

Author

David Schaefer

Date

June 2015

16.69.2.2 getdata()

```
generic, public mo_ncdf::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	<i>integer(i4) :: start(:), cnt(:), stride(:), map(:)</i>	
out	<i>integer(i4)/real(sp)/real(dp), allocatable, dimension(((:)/(,:) /(:, :, :)/(:, :, :, :)) :: values</i>	

Author

David Schaefer

Date

June 2015

16.69.2.3 getdata1df32()

```
procedure, private mo_ncdf::ncdimension::getdata1df32 ( ) [private]
```

16.69.2.4 getdata1df64()

```
procedure, private mo_ncdf::ncdimension::getdata1df64 ( ) [private]
```

16.69.2.5 getdata1di16()

```
procedure, private mo_ncdf::ncdimension::getdata1di16 ( ) [private]
```

16.69.2.6 getdata1di32()

```
procedure, private mo_ncdf::ncdimension::getdata1di32 ( ) [private]
```

16.69.2.7 getdata1di64()

```
procedure, private mo_netcdf::ncdimension::getdata1di64 ( ) [private]
```

16.69.2.8 getdata1di8()

```
procedure, private mo_netcdf::ncdimension::getdata1di8 ( ) [private]
```

16.69.2.9 getdata2df32()

```
procedure, private mo_netcdf::ncdimension::getdata2df32 ( ) [private]
```

16.69.2.10 getdata2df64()

```
procedure, private mo_netcdf::ncdimension::getdata2df64 ( ) [private]
```

16.69.2.11 getdata2di16()

```
procedure, private mo_netcdf::ncdimension::getdata2di16 ( ) [private]
```

16.69.2.12 getdata2di32()

```
procedure, private mo_netcdf::ncdimension::getdata2di32 ( ) [private]
```

16.69.2.13 getdata2di64()

```
procedure, private mo_netcdf::ncdimension::getdata2di64 ( ) [private]
```

16.69.2.14 getdata2di8()

```
procedure, private mo_netcdf::ncdimension::getdata2di8 ( ) [private]
```

16.69.2.15 getdata3df32()

```
procedure, private mo_netcdf::ncdimension::getdata3df32 ( ) [private]
```

16.69.2.16 getdata3df64()

```
procedure, private mo_ncdf::ncdimension::getdata3df64 ( ) [private]
```

16.69.2.17 getdata3di16()

```
procedure, private mo_ncdf::ncdimension::getdata3di16 ( ) [private]
```

16.69.2.18 getdata3di32()

```
procedure, private mo_ncdf::ncdimension::getdata3di32 ( ) [private]
```

16.69.2.19 getdata3di64()

```
procedure, private mo_ncdf::ncdimension::getdata3di64 ( ) [private]
```

16.69.2.20 getdata3di8()

```
procedure, private mo_ncdf::ncdimension::getdata3di8 ( ) [private]
```

16.69.2.21 getdata4df32()

```
procedure, private mo_ncdf::ncdimension::getdata4df32 ( ) [private]
```

16.69.2.22 getdata4df64()

```
procedure, private mo_ncdf::ncdimension::getdata4df64 ( ) [private]
```

16.69.2.23 getdata4di16()

```
procedure, private mo_ncdf::ncdimension::getdata4di16 ( ) [private]
```

16.69.2.24 getdata4di32()

```
procedure, private mo_ncdf::ncdimension::getdata4di32 ( ) [private]
```

16.69.2.25 getdata4di64()

```
procedure, private mo_netcdf::ncdimension::getdata4di64 ( ) [private]
```

16.69.2.26 getdata4di8()

```
procedure, private mo_netcdf::ncdimension::getdata4di8 ( ) [private]
```

16.69.2.27 getdata5df32()

```
procedure, private mo_netcdf::ncdimension::getdata5df32 ( ) [private]
```

16.69.2.28 getdata5df64()

```
procedure, private mo_netcdf::ncdimension::getdata5df64 ( ) [private]
```

16.69.2.29 getdata5di16()

```
procedure, private mo_netcdf::ncdimension::getdata5di16 ( ) [private]
```

16.69.2.30 getdata5di32()

```
procedure, private mo_netcdf::ncdimension::getdata5di32 ( ) [private]
```

16.69.2.31 getdata5di64()

```
procedure, private mo_netcdf::ncdimension::getdata5di64 ( ) [private]
```

16.69.2.32 getdata5di8()

```
procedure, private mo_netcdf::ncdimension::getdata5di8 ( ) [private]
```

16.69.2.33 getdatascalarf32()

```
procedure, private mo_netcdf::ncdimension::getdatascalarf32 ( ) [private]
```

16.69.2.34 getdatascalarf64()

```
procedure, private mo_ncdf::ncdimension::getdatascalarf64 ( ) [private]
```

16.69.2.35 getdatascalari16()

```
procedure, private mo_ncdf::ncdimension::getdatascalari16 ( ) [private]
```

16.69.2.36 getdatascalari32()

```
procedure, private mo_ncdf::ncdimension::getdatascalari32 ( ) [private]
```

16.69.2.37 getdatascalari64()

```
procedure, private mo_ncdf::ncdimension::getdatascalari64 ( ) [private]
```

16.69.2.38 getdatascalari8()

```
procedure, private mo_ncdf::ncdimension::getdatascalari8 ( ) [private]
```

16.69.2.39 getdimensions()

```
procedure, public mo_ncdf::ncdimension::getdimensions ( )
```

Retrieve the variable dimensions.

Return the ids of the dimensions associated with variable.

16.69.2.40 getdtype()

```
procedure, public mo_ncdf::ncdimension::getdtype ( )
```

Retrieve the variable data type.

Return the encoded data type of the variable. Data type encodeings "f32" -> NF90_FLOAT "f64" -> NF90_DOUBLE "i8" -> NF90_BYT "i16" -> NF90_SHORT "i32" -> NF90_INT "i64" -> NF90_INT64

16.69.2.41 getfillvalue()

```
generic, public mo_ncdf::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 termination.

Parameters

out	<i>integer(i4)/real(sp)/real(dp) :: fvalue</i>	
-----	--	--

Author

David Schaefer

Date

June 2015

16.69.2.42 getname()

```
procedure, public mo_netcdf::ncdimension::getname ( )
```

16.69.2.43 getnodimensions()

```
procedure, public mo_netcdf::ncdimension::getnodimensions ( )
```

Retrieve the number of dimensions.

Return the number of dimensions associated with variable

16.69.2.44 getshape()

```
procedure, public mo_netcdf::ncdimension::getshape ( )
```

Retrieve the shape of the variable.

Return the shape of the variable.

16.69.2.45 getvariableattributechar()

```
procedure, private mo_netcdf::ncdimension::getvariableattributechar ( ) [private]
```

16.69.2.46 getvariableattributef32()

```
procedure, private mo_netcdf::ncdimension::getvariableattributef32 ( ) [private]
```

16.69.2.47 getvariableattributef64()

```
procedure, private mo_netcdf::ncdimension::getvariableattributef64 ( ) [private]
```

16.69.2.48 getvariableattributei16()

```
procedure, private mo_netcdf::ncdimension::getvariableattributei16 ( ) [private]
```

16.69.2.49 getvariableattributei32()

```
procedure, private mo_ncdf::ncdimension::getvariableattributei32 ( ) [private]
```

16.69.2.50 getvariableattributei64()

```
procedure, private mo_ncdf::ncdimension::getvariableattributei64 ( ) [private]
```

16.69.2.51 getvariableattributei8()

```
procedure, private mo_ncdf::ncdimension::getvariableattributei8 ( ) [private]
```

16.69.2.52 getvariablefillvaluef32()

```
procedure, private mo_ncdf::ncdimension::getvariablefillvaluef32 ( ) [private]
```

16.69.2.53 getvariablefillvaluef64()

```
procedure, private mo_ncdf::ncdimension::getvariablefillvaluef64 ( ) [private]
```

16.69.2.54 getvariablefillvaluei16()

```
procedure, private mo_ncdf::ncdimension::getvariablefillvaluei16 ( ) [private]
```

16.69.2.55 getvariablefillvaluei32()

```
procedure, private mo_ncdf::ncdimension::getvariablefillvaluei32 ( ) [private]
```

16.69.2.56 getvariablefillvaluei64()

```
procedure, private mo_ncdf::ncdimension::getvariablefillvaluei64 ( ) [private]
```

16.69.2.57 getvariablefillvaluei8()

```
procedure, private mo_ncdf::ncdimension::getvariablefillvaluei8 ( ) [private]
```

16.69.2.58 hasattribute()

```
procedure, public mo_netcdf::ncdimension::hasattribute ( )
```

Check if attribute exists.

Returns true if an attribute with the given name exists, false otherwise.

16.69.2.59 initncvariable()

```
procedure, public mo_netcdf::ncdimension::initncvariable ( )
```

16.69.2.60 isunlimited()

```
procedure, public mo_netcdf::ncdimension::isunlimited ( )
```

Check if the variable is unlimited.

Returns true if the variable has an unlimited dimension, false otherwise.

Returns

"logical"

Author

David Schaefer

Date

June 2015

16.69.2.61 setattribute()

```
generic, public mo_netcdf::ncdimension::setattribute ( )
```

Create a new variable attribute.

Create a new variable attribute from given name and value. A write error results in abrupt program termination.

Parameters

in	<i>character(*) :: name</i>	
in	<i>character(*)/integer(i4)/real(sp)/real(dp) :: value</i>	

Author

David Schaefer

Date

June 2015

16.69.2.62 setdata()

```
generic, public mo_netcdf::ncdimension::setdata ( )
```

Write data to variable.

Write the given data into the variable at an optionally given position. All optional arguments to the nf90_put_var function are supported. A write error will result in abrupt program termination.

Parameters

in	<i>integer(i4)/real(sp)/real(dp), dimension((():/(:,:)/(:,:,:)/(:,:,:,:)/(:,:,:,:,:)) :: values</i>	
in	<i>integer(i4) :: start(:), cnt(:), stride(:), map(:)</i>	

Author

David Schaefer

Date

June 2015

16.69.2.63 setdata1df32()

```
procedure, private mo_netcdf::ncdimension::setdata1df32 ( ) [private]
```

16.69.2.64 setdata1df64()

```
procedure, private mo_netcdf::ncdimension::setdata1df64 ( ) [private]
```

16.69.2.65 setdata1di16()

```
procedure, private mo_netcdf::ncdimension::setdata1di16 ( ) [private]
```

16.69.2.66 setdata1di32()

```
procedure, private mo_netcdf::ncdimension::setdata1di32 ( ) [private]
```

16.69.2.67 setdata1di64()

```
procedure, private mo_netcdf::ncdimension::setdata1di64 ( ) [private]
```

16.69.2.68 setdata1di8()

```
procedure, private mo_netcdf::ncdimension::setdata1di8 ( ) [private]
```

16.69.2.69 setdata2df32()

```
procedure, private mo_netcdf::ncdimension::setdata2df32 ( ) [private]
```

16.69.2.70 setdata2df64()

```
procedure, private mo_netcdf::ncdimension::setdata2df64 ( ) [private]
```

16.69.2.71 setdata2di16()

```
procedure, private mo_netcdf::ncdimension::setdata2di16 ( ) [private]
```

16.69.2.72 setdata2di32()

```
procedure, private mo_netcdf::ncdimension::setdata2di32 ( ) [private]
```

16.69.2.73 setdata2di64()

```
procedure, private mo_netcdf::ncdimension::setdata2di64 ( ) [private]
```

16.69.2.74 setdata2di8()

```
procedure, private mo_netcdf::ncdimension::setdata2di8 ( ) [private]
```

16.69.2.75 setdata3df32()

```
procedure, private mo_netcdf::ncdimension::setdata3df32 ( ) [private]
```

16.69.2.76 setdata3df64()

```
procedure, private mo_netcdf::ncdimension::setdata3df64 ( ) [private]
```

16.69.2.77 setdata3di16()

```
procedure, private mo_ncdf::ncdimension::setdata3di16 ( ) [private]
```

16.69.2.78 setdata3di32()

```
procedure, private mo_ncdf::ncdimension::setdata3di32 ( ) [private]
```

16.69.2.79 setdata3di64()

```
procedure, private mo_ncdf::ncdimension::setdata3di64 ( ) [private]
```

16.69.2.80 setdata3di8()

```
procedure, private mo_ncdf::ncdimension::setdata3di8 ( ) [private]
```

16.69.2.81 setdata4df32()

```
procedure, private mo_ncdf::ncdimension::setdata4df32 ( ) [private]
```

16.69.2.82 setdata4df64()

```
procedure, private mo_ncdf::ncdimension::setdata4df64 ( ) [private]
```

16.69.2.83 setdata4di16()

```
procedure, private mo_ncdf::ncdimension::setdata4di16 ( ) [private]
```

16.69.2.84 setdata4di32()

```
procedure, private mo_ncdf::ncdimension::setdata4di32 ( ) [private]
```

16.69.2.85 setdata4di64()

```
procedure, private mo_ncdf::ncdimension::setdata4di64 ( ) [private]
```

16.69.2.86 setdata4di8()

```
procedure, private mo_netcdf::ncdimension::setdata4di8 ( ) [private]
```

16.69.2.87 setdata5df32()

```
procedure, private mo_netcdf::ncdimension::setdata5df32 ( ) [private]
```

16.69.2.88 setdata5df64()

```
procedure, private mo_netcdf::ncdimension::setdata5df64 ( ) [private]
```

16.69.2.89 setdata5di16()

```
procedure, private mo_netcdf::ncdimension::setdata5di16 ( ) [private]
```

16.69.2.90 setdata5di32()

```
procedure, private mo_netcdf::ncdimension::setdata5di32 ( ) [private]
```

16.69.2.91 setdata5di64()

```
procedure, private mo_netcdf::ncdimension::setdata5di64 ( ) [private]
```

16.69.2.92 setdata5di8()

```
procedure, private mo_netcdf::ncdimension::setdata5di8 ( ) [private]
```

16.69.2.93 setdatascalarf32()

```
procedure, private mo_netcdf::ncdimension::setdatascalarf32 ( ) [private]
```

16.69.2.94 setdatascalarf64()

```
procedure, private mo_netcdf::ncdimension::setdatascalarf64 ( ) [private]
```

16.69.2.95 setdatascalari16()

```
procedure, private mo_ncdf::ncdimension::setdatascalari16 ( ) [private]
```

16.69.2.96 setdatascalari32()

```
procedure, private mo_ncdf::ncdimension::setdatascalari32 ( ) [private]
```

16.69.2.97 setdatascalari64()

```
procedure, private mo_ncdf::ncdimension::setdatascalari64 ( ) [private]
```

16.69.2.98 setdatascalari8()

```
procedure, private mo_ncdf::ncdimension::setdatascalari8 ( ) [private]
```

16.69.2.99 setfillvalue()

generic, public mo_ncdf::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	<i>integer(i4)/real(sp)/real(dp) :: fvalue</i>	
----	--	--

Author

David Schaefer

Date

June 2015

16.69.2.100 setvariableattributechar()

```
procedure, private mo_ncdf::ncdimension::setvariableattributechar ( ) [private]
```

16.69.2.101 setvariableattributef32()

```
procedure, private mo_netcdf::ncdimension::setvariableattributef32 ( ) [private]
```

16.69.2.102 setvariableattributef64()

```
procedure, private mo_netcdf::ncdimension::setvariableattributef64 ( ) [private]
```

16.69.2.103 setvariableattributei16()

```
procedure, private mo_netcdf::ncdimension::setvariableattributei16 ( ) [private]
```

16.69.2.104 setvariableattributei32()

```
procedure, private mo_netcdf::ncdimension::setvariableattributei32 ( ) [private]
```

16.69.2.105 setvariableattributei64()

```
procedure, private mo_netcdf::ncdimension::setvariableattributei64 ( ) [private]
```

16.69.2.106 setvariableattributei8()

```
procedure, private mo_netcdf::ncdimension::setvariableattributei8 ( ) [private]
```

16.69.2.107 setvariablefillvaluef32()

```
procedure, private mo_netcdf::ncdimension::setvariablefillvaluef32 ( ) [private]
```

16.69.2.108 setvariablefillvaluef64()

```
procedure, private mo_netcdf::ncdimension::setvariablefillvaluef64 ( ) [private]
```

16.69.2.109 setvariablefillvaluei16()

```
procedure, private mo_netcdf::ncdimension::setvariablefillvaluei16 ( ) [private]
```

16.69.2.110 setvariablefillvaluei32()

```
procedure, private mo_ncdf::ncdimension::setvariablefillvaluei32 ( ) [private]
```

16.69.2.111 setvariablefillvaluei64()

```
procedure, private mo_ncdf::ncdimension::setvariablefillvaluei64 ( ) [private]
```

16.69.2.112 setvariablefillvaluei8()

```
procedure, private mo_ncdf::ncdimension::setvariablefillvaluei8 ( ) [private]
```

16.69.3 Member Data Documentation

16.69.3.1 dimension [1/2]

```
integer(i4) mo_ncdf::ncdimension::dimension
```

16.69.3.2 dimension [2/2]

```
type(ncdataset) mo_ncdf::ncdimension::dimension
```

16.69.3.3 id

```
integer(i4) mo_ncdf::ncdimension::id
```

16.69.3.4 netcdf

```
integer(i4) mo_ncdf::ncdimension::netcdf
```

16.69.3.5 parent

```
type(ncdataset) mo_ncdf::ncdimension::parent
```

16.69.3.6 s

```
type(ncdataset) mo_ncdf::ncdimension::s
```

16.69.3.7 the [1/2]

```
integer(i4) mo_netcdf::ncdimension::the
```

16.69.3.8 the [2/2]

```
type(ncdataset) mo_netcdf::ncdimension::the
```

The documentation for this type was generated from the following file:

- [mo_netcdf.f90](#)

16.70 mo_netcdf::ncvariable Interface Reference

The documentation for this interface was generated from the following file:

- [mo_netcdf.f90](#)

16.71 mo_utils::ne Interface Reference

Public Member Functions

- elemental pure logical function [notequal_sp](#) (a, b)
- elemental pure logical function [notequal_dp](#) (a, b)

16.71.1 Member Function/Subroutine Documentation

16.71.1.1 notequal_dp()

```
elemental pure logical function mo_utils::ne::notequal_dp (
    real(dp), intent(in) a,
    real(dp), intent(in) b )
```

16.71.1.2 notequal_sp()

```
elemental pure logical function mo_utils::ne::notequal_sp (
    real(sp), intent(in) a,
    real(sp), intent(in) b )
```

The documentation for this interface was generated from the following file:

- [mo_utils.f90](#)

16.72 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)

16.72.1 Member Function/Subroutine Documentation

16.72.1.1 [d_nearless\(\)](#)

```
real(kind = dp) function mo_orderpack::nearless::d_nearless (
    real(kind = dp), intent(in) XVAL )
```

16.72.1.2 [i_nearless\(\)](#)

```
integer(kind = i4) function mo_orderpack::nearless::i_nearless (
    integer(kind = i4), intent(in) XVAL )
```

16.72.1.3 [r_nearless\(\)](#)

```
real(kind = sp) function mo_orderpack::nearless::r_nearless (
    real(kind = sp), intent(in) XVAL )
```

The documentation for this interface was generated from the following file:

- [mo_orderpack.f90](#)

16.73 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)

16.73.1 Detailed Description

Calculates the number of neighboring dominating values, a measure for spatial dissimilarity.

$NNDV = 1 - \frac{\sum(\text{abs}(\text{dominating_neighbors}(\text{mat1}) - \text{dominating_neighbors}(\text{mat2})))}{\text{count}(\text{mask})}$

$\text{dominating_neighbors}(\text{mat1}) = \text{comparison if pixel is larger than its neighbouring values}$

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 subtraction 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 mismatching.

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, FTT TFF FFF |
       | FFF TFT 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 optional 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	<i>real(sp/dp), dimension(:, :) :: mat1</i>	2D-array with input numbers
in	<i>real(sp/dp), dimension(:, :) :: mat2</i>	2D-array with input numbers
in	<i>logical, dimension(:, :, optional :: mask</i>	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	<i>logical :: valid</i>	indicates if the function could determine a valid value result can be invalid 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

16.73.2 Member Function/Subroutine Documentation

16.73.2.1 nndv_dp()

```
real(dp) function mo_spatialsimilarity::nndv::nndv_dp (
    real(dp), dimension(:, :), intent(in) mat1,
    real(dp), dimension(:, :), intent(in) mat2,
    logical, dimension(:, :), intent(in), optional mask,
    logical, intent(out), optional valid )
```

16.73.2.2 nndv_sp()

```
real(sp) function mo_spatialsimilarity::nndv::nndv_sp (
    real(sp), dimension(:, :), intent(in) mat1,
    real(sp), dimension(:, :), intent(in) mat2,
    logical, dimension(:, :), intent(in), optional mask,
    logical, intent(out), optional valid )
```

The documentation for this interface was generated from the following file:

- [mo_spatialsimilarity.f90](#)

16.74 mo_utils::notequal Interface Reference

Public Member Functions

- elemental pure logical function [notequal_sp](#) (a, b)
- elemental pure logical function [notequal_dp](#) (a, b)

16.74.1 Member Function/Subroutine Documentation

16.74.1.1 notequal_dp()

```
elemental pure logical function mo_utils::notequal::notequal_dp (
    real(dp), intent(in) a,
    real(dp), intent(in) b )
```

16.74.1.2 notequal_sp()

```
elemental pure logical function mo_utils::notequal::notequal_sp (
    real(sp), intent(in) a,
    real(sp), intent(in) b )
```

The documentation for this interface was generated from the following file:

- [mo_utils.f90](#)

16.75 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)

16.75.1 Member Function/Subroutine Documentation

16.75.1.1 nse_dp_1d()

```
real(dp) function mo_errormeasures::nse::nse_dp_1d (
    real(dp), dimension(:), intent(in) x,
    real(dp), dimension(:), intent(in) y,
    logical, dimension(:), intent(in), optional mask )
```

16.75.1.2 nse_dp_2d()

```
real(dp) function mo_errormeasures::nse::nse_dp_2d (
    real(dp), dimension(:, :), intent(in) x,
    real(dp), dimension(:, :), intent(in) y,
    logical, dimension(:, :, :), intent(in), optional mask )
```

16.75.1.3 nse_dp_3d()

```
real(dp) function mo_errormeasures::nse::nse_dp_3d (
    real(dp), dimension(:, :, :, :), intent(in) x,
    real(dp), dimension(:, :, :, :), intent(in) y,
    logical, dimension(:, :, :, :), intent(in), optional mask )
```

16.75.1.4 nse_sp_1d()

```
real(sp) function mo_errormeasures::nse::nse_sp_1d (
    real(sp), dimension(:), intent(in) x,
    real(sp), dimension(:), intent(in) y,
    logical, dimension(:), intent(in), optional mask )
```

16.75.1.5 nse_sp_2d()

```
real(sp) function mo_errormeasures::nse::nse_sp_2d (
    real(sp), dimension(:, :), intent(in) x,
    real(sp), dimension(:, :), intent(in) y,
    logical, dimension(:, :, :), intent(in), optional mask )
```

16.75.1.6 nse_sp_3d()

```
real(sp) function mo_errormeasures::nse::nse_sp_3d (
    real(sp), dimension(:, :, :, :), intent(in) x,
    real(sp), dimension(:, :, :, :), intent(in) y,
    logical, dimension(:, :, :, :), intent(in), optional mask )
```

The documentation for this interface was generated from the following file:

- [mo_errormeasures.f90](#)

16.76 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)

16.76.1 Detailed Description

Convert to string.

Convert a number or logical to a string with an optional format.

Parameters

in	<i>integer(i4/i8)/real(sp/dp)/logical :: num</i>	Number or logical
in	<i>character(len=*)</i> , optional :: 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
 Output 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

16.76.2 Member Function/Subroutine Documentation

16.76.2.1 dp2str()

```
pure character(len = 32) function mo_string_utils::num2str::dp2str (
    real(dp), intent(in) rr,
    character(len = *), intent(in), optional form )
```

16.76.2.2 i42str()

```
pure character(len = 10) function mo_string_utils::num2str::i42str (
    integer(i4), intent(in) nn,
    character(len = *), intent(in), optional form )
```

16.76.2.3 i82str()

```
pure character(len = 20) function mo_string_utils::num2str::i82str (
    integer(i8), intent(in) nn,
    character(len = *), intent(in), optional form )
```

16.76.2.4 log2str()

```
pure character(len = 10) function mo_string_utils::num2str::log2str (
    logical, intent(in) ll,
    character(len = *), intent(in), optional form )
```

16.76.2.5 sp2str()

```
pure character(len = 32) function mo_string_utils::num2str::sp2str (
    real(sp), intent(in) rr,
    character(len = *), intent(in), optional form )
```

The documentation for this interface was generated from the following file:

- [mo_string_utils.f90](#)

16.77 mo_string_utils::numarray2str Interface Reference

Convert to string.

Public Member Functions

- character(len=size(arr)) function [i4array2str](#) (arr)

16.77.1 Detailed Description

Convert to string.

Convert a array of numbers or logicals to a string.

Parameters

in	<i>integer(i4/i8)/real(sp/dp)/logical :: num(:)</i>	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

16.77.2 Member Function/Subroutine Documentation

16.77.2.1 i4array2str()

```
character(len = size(arr)) function mo_string_utils::numarray2str::i4array2str (
    integer(i4), dimension(:), intent(in) arr )
```

The documentation for this interface was generated from the following file:

- [mo_string_utils.f90](#)

16.78 mo_optimization_utils::objective_interface Interface Reference

Public Member Functions

- real(dp) function [objective_interface](#) (parameterset, eval, arg1, arg2, arg3)

16.78.1 Constructor & Destructor Documentation

16.78.1.1 objective_interface()

```
real(dp) function mo_optimization_utils::objective_interface::objective_interface (
    real(dp), dimension(:), intent(in) parameterset,
    procedure(eval\_interface), intent(in), pointer eval,
    real(dp), intent(in), optional arg1,
    real(dp), intent(out), optional arg2,
    real(dp), intent(out), optional arg3 )
```

References mo_kind::dp.

The documentation for this interface was generated from the following file:

- [mo_optimization_utils.f90](#)

16.79 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)

16.79.1 Member Function/Subroutine Documentation

16.79.1.1 d_median()

```
real(kind = dp) function mo_orderpack::omedian::d_median (
    real(kind = dp), dimension (:), intent(in) XDONT )
```

16.79.1.2 i_median()

```
integer(kind = i4) function mo_orderpack::omedian::i_median (
    integer(kind = i4), dimension (:), intent(in) XDONT )
```

16.79.1.3 r_median()

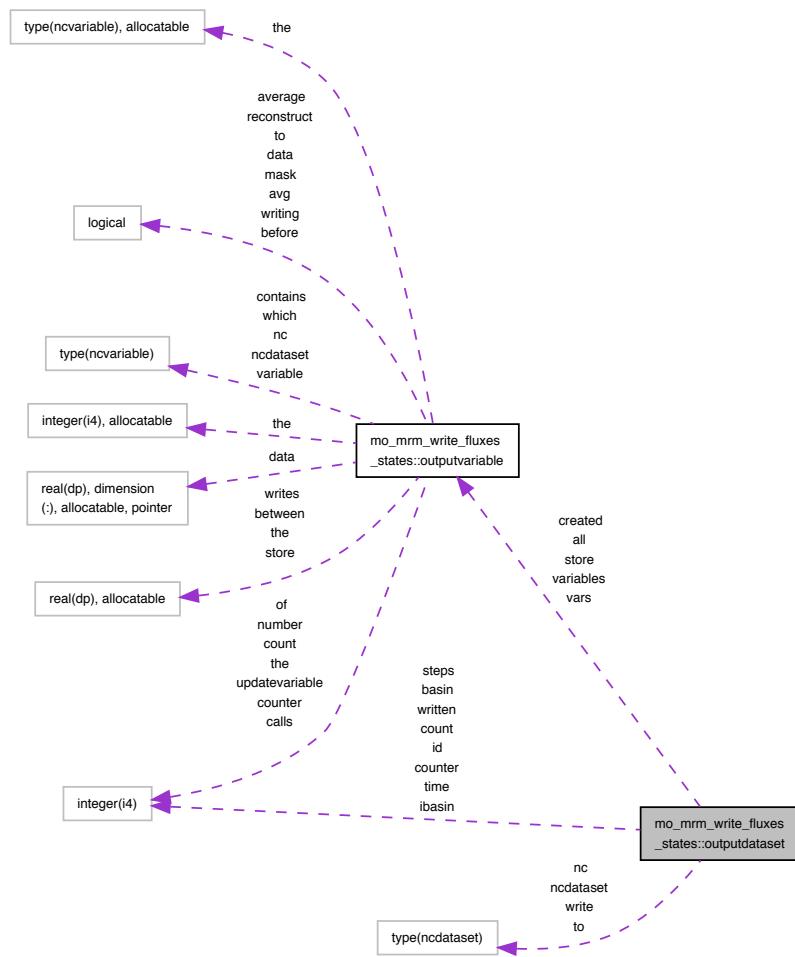
```
real(kind = sp) function mo_orderpack::omedian::r_median (
    real(kind = sp), dimension (:), intent(in) XDONT )
```

The documentation for this interface was generated from the following file:

- [mo_orderpack.f90](#)

16.80 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**

16.80.1 Member Function/Subroutine Documentation

16.80.1.1 **close()**

```
procedure, public mo_mrm_write_fluxes_states::outputdataset::close ( )
```

16.80.1.2 **updatedataset()**

```
procedure, public mo_mrm_write_fluxes_states::outputdataset::updatedataset ( )
```

16.80.1.3 **writetimestep()**

```
procedure, public mo_mrm_write_fluxes_states::outputdataset::writetimestep ( )
```

16.80.2 Member Data Documentation

16.80.2.1 **all**

```
type(outputvariable), allocatable mo_mrm_write_fluxes_states::outputdataset::all
```

16.80.2.2 **basin**

```
integer(i4) mo_mrm_write_fluxes_states::outputdataset::basin
```

16.80.2.3 count

```
integer(i4) mo_mrm_write_fluxes_states::outputdataset::count
```

16.80.2.4 counter

```
integer(i4) mo_mrm_write_fluxes_states::outputdataset::counter = 0
```

16.80.2.5 created

```
type(outputvariable), dimension (dynamic), allocatable mo_mrm_write_fluxes_states::outputdataset:::created
```

16.80.2.6 ibasin

```
integer(i4) mo_mrm_write_fluxes_states::outputdataset::ibasin
```

16.80.2.7 id

```
integer(i4) mo_mrm_write_fluxes_states::outputdataset::id
```

16.80.2.8 nc

```
type(ncdataset) mo_mrm_write_fluxes_states::outputdataset::nc
```

16.80.2.9 ncdataset

```
type(ncdataset) mo_mrm_write_fluxes_states::outputdataset::ncdataset
```

16.80.2.10 steps

```
integer(i4) mo_mrm_write_fluxes_states::outputdataset::steps
```

16.80.2.11 store

```
type(outputvariable), allocatable mo_mrm_write_fluxes_states::outputdataset::store
```

16.80.2.12 time

```
integer(i4) mo_mrm_write_fluxes_states::outputdataset::time
```

16.80.2.13 to

```
type(ncdataset) mo_mrm_write_fluxes_states::outputdataset::to
```

16.80.2.14 variables

```
type(outputvariable), allocatable mo_mrm_write_fluxes_states::outputdataset::variables
```

16.80.2.15 vars

```
type(outputvariable), dimension(:), allocatable mo_mrm_write_fluxes_states::outputdataset::vars
```

16.80.2.16 write

```
type(ncdataset) mo_mrm_write_fluxes_states::outputdataset::write
```

16.80.2.17 written

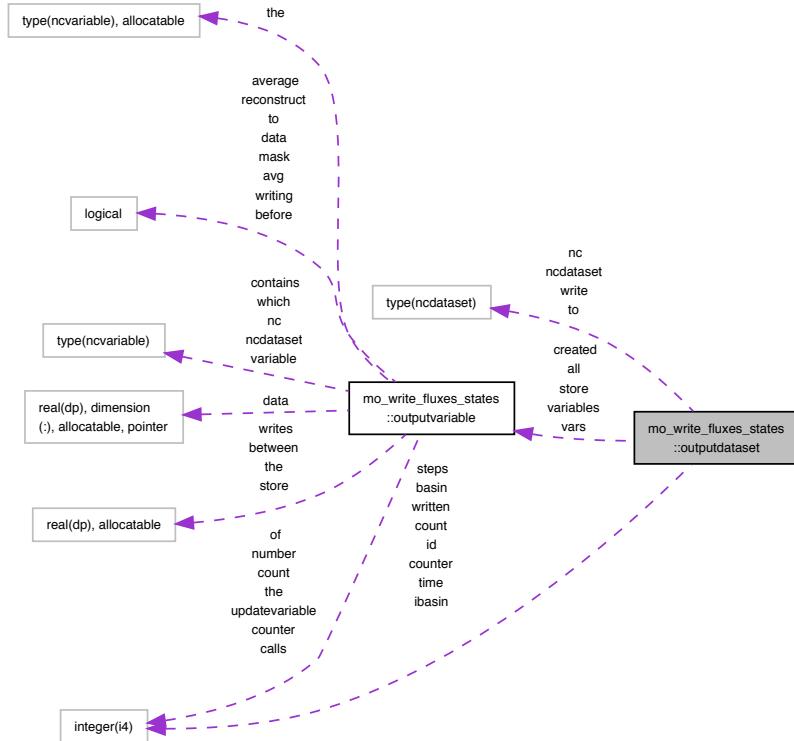
```
integer(i4) mo_mrm_write_fluxes_states::outputdataset::written
```

The documentation for this interface was generated from the following file:

- [mo_mrm_write_fluxes_states.f90](#)

16.81 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](#)

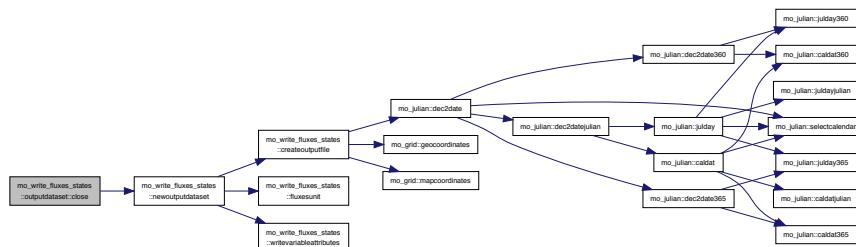
16.81.1 Member Function/Subroutine Documentation

16.81.1.1 [close\(\)](#)

```
procedure, public mo_write_fluxes_states::outputdataset::close ( )
```

References [mo_write_fluxes_states::newoutputdataset\(\)](#).

Here is the call graph for this function:



16.81.1.2 [updatedataset\(\)](#)

```
procedure, public mo_write_fluxes_states::outputdataset::updatedataset ( )
```

16.81.1.3 [writetimestep\(\)](#)

```
procedure, public mo_write_fluxes_states::outputdataset::writetimestep ( )
```

16.81.2 Member Data Documentation

16.81.2.1 [all](#)

```
type(outputvariable), allocatable mo_write_fluxes_states::outputdataset::all
```

16.81.2.2 [basin](#)

```
integer(i4) mo_write_fluxes_states::outputdataset::basin
```

16.81.2.3 count

```
integer(i4) mo_write_fluxes_states::outputdataset::count
```

16.81.2.4 counter

```
integer(i4) mo_write_fluxes_states::outputdataset::counter = 0
```

16.81.2.5 created

```
type(outputvariable), dimension (dynamic), allocatable mo_write_fluxes_states::outputdataset::created
```

16.81.2.6 ibasin

```
integer(i4) mo_write_fluxes_states::outputdataset::ibasin
```

16.81.2.7 id

```
integer(i4) mo_write_fluxes_states::outputdataset::id
```

16.81.2.8 nc

```
type(ncdataset) mo_write_fluxes_states::outputdataset::nc
```

16.81.2.9 ncdataset

```
type(ncdataset) mo_write_fluxes_states::outputdataset::ncdataset
```

16.81.2.10 steps

```
integer(i4) mo_write_fluxes_states::outputdataset::steps
```

16.81.2.11 store

```
type(outputvariable), allocatable mo_write_fluxes_states::outputdataset::store
```

16.81.2.12 time

```
integer(i4) mo_write_fluxes_states::outputdataset::time
```

16.81.2.13 to

```
type(ncdataset) mo_write_fluxes_states::outputdataset::to
```

16.81.2.14 variables

```
type(outputvariable), allocatable mo_write_fluxes_states::outputdataset::variables
```

16.81.2.15 vars

```
type(outputvariable), dimension(:), allocatable mo_write_fluxes_states::outputdataset::vars
```

16.81.2.16 write

```
type(ncdataset) mo_write_fluxes_states::outputdataset::write
```

16.81.2.17 written

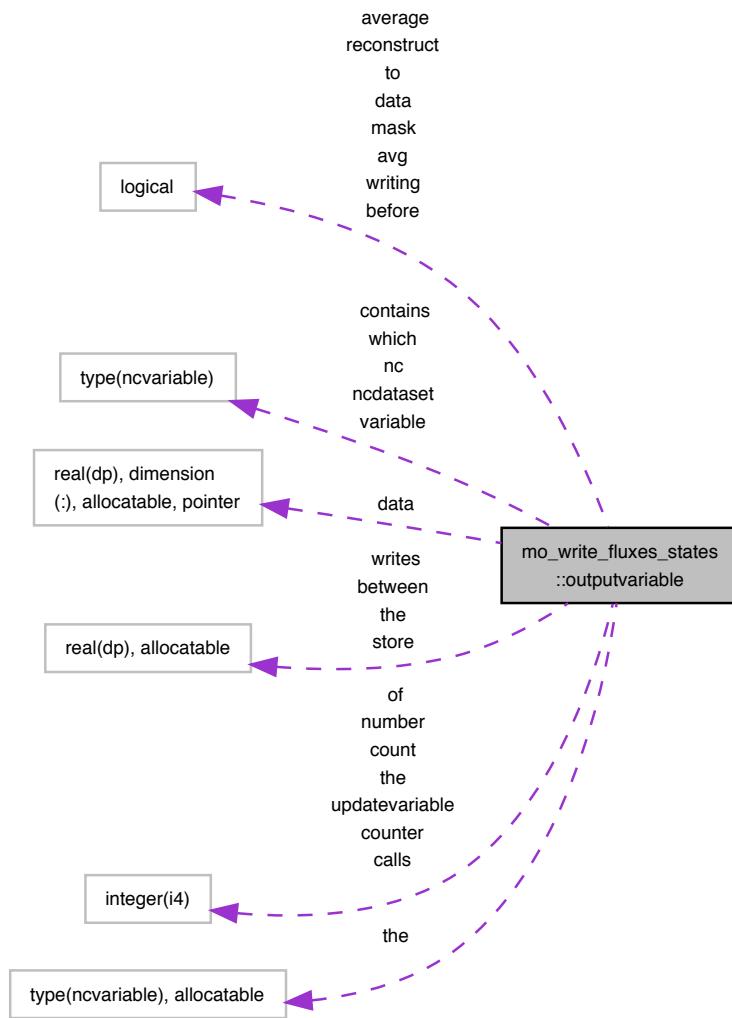
```
integer(i4) mo_write_fluxes_states::outputdataset::written
```

The documentation for this interface was generated from the following file:

- [mo_write_fluxes_states.f90](#)

16.82 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**

16.82.1 Member Function/Subroutine Documentation

16.82.1.1 **updatevariable()**

```
procedure, public mo_write_fluxes_states::outputvariable::updatevariable ( )
```

16.82.1.2 **writevariabletimestep()**

```
procedure, public mo_write_fluxes_states::outputvariable::writevariabletimestep ( )
```

16.82.2 Member Data Documentation

16.82.2.1 **average**

```
logical mo_write_fluxes_states::outputvariable::average
```

16.82.2.2 **avg**

```
logical mo_write_fluxes_states::outputvariable::avg = .false.
```

16.82.2.3 before

```
logical mo_write_fluxes_states::outputvariable::before
```

16.82.2.4 between

```
real(dp), allocatable mo_write_fluxes_states::outputvariable::between
```

16.82.2.5 calls

```
integer(i4) mo_write_fluxes_states::outputvariable::calls
```

16.82.2.6 contains

```
type(ncvariable) mo_write_fluxes_states::outputvariable::contains
```

16.82.2.7 count

```
integer(i4) mo_write_fluxes_states::outputvariable::count
```

16.82.2.8 counter

```
integer(i4) mo_write_fluxes_states::outputvariable::counter = 0
```

16.82.2.9 data [1/2]

```
logical, dimension(:), allocatable, pointer mo_write_fluxes_states::outputvariable::data
```

16.82.2.10 data [2/2]

```
real(dp), dimension(:), allocatable, pointer mo_write_fluxes_states::outputvariable::data
```

16.82.2.11 mask

```
logical, dimension(:, :), pointer mo_write_fluxes_states::outputvariable::mask
```

16.82.2.12 nc

```
type(ncvariable) mo_write_fluxes_states::outputvariable::nc
```

16.82.2.13 ncdataset

```
type(ncvariable) mo_write_fluxes_states::outputvariable::ncdataset
```

16.82.2.14 number

```
integer(i4) mo_write_fluxes_states::outputvariable::number
```

16.82.2.15 of

```
integer(i4) mo_write_fluxes_states::outputvariable::of
```

16.82.2.16 reconstruct

```
logical, pointer mo_write_fluxes_states::outputvariable::reconstruct
```

16.82.2.17 store

```
real(dp), allocatable mo_write_fluxes_states::outputvariable::store
```

16.82.2.18 the [1/3]

```
type(ncvariable), allocatable mo_write_fluxes_states::outputvariable::the
```

16.82.2.19 the [2/3]

```
real(dp), allocatable mo_write_fluxes_states::outputvariable::the
```

16.82.2.20 the [3/3]

```
integer(i4), allocatable mo_write_fluxes_states::outputvariable::the
```

16.82.2.21 to

```
logical, pointer mo_write_fluxes_states::outputvariable::to
```

16.82.2.22 updatevariable

```
integer(i4), public mo_write_fluxes_states::outputvariable::updatevariable
```

16.82.2.23 variable

```
type(ncvariable) mo_write_fluxes_states::outputvariable::variable
```

16.82.2.24 which

```
type(ncvariable) mo_write_fluxes_states::outputvariable::which
```

16.82.2.25 writes

```
real(dp), allocatable mo_write_fluxes_states::outputvariable::writes
```

16.82.2.26 writing

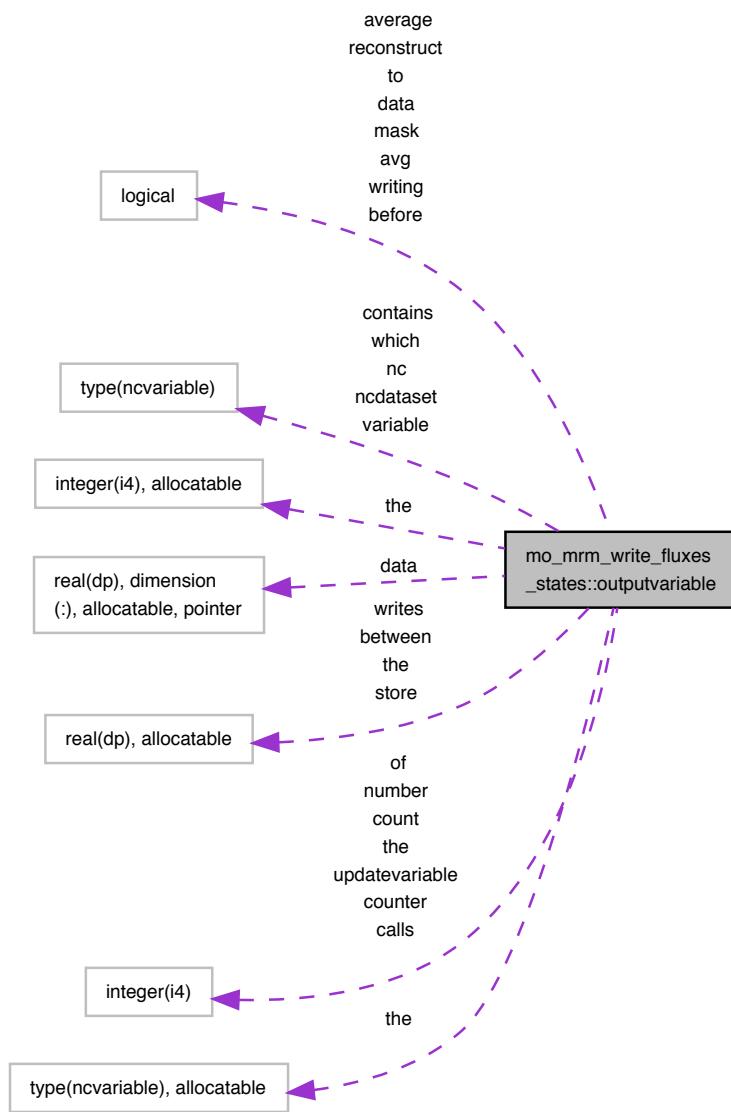
```
logical mo_write_fluxes_states::outputvariable::writing
```

The documentation for this interface was generated from the following file:

- [mo_write_fluxes_states.f90](#)

16.83 mo_mrm_write_fluxes_states::outputvariable Interface Reference

Collaboration diagram for mo_mrm_write_fluxes_states::outputvariable:



Public Member Functions

- procedure, public `updatevariable`
- procedure, public `writevariabletimestep`

Public Attributes

- `type(ncvariable) nc`
- `type(ncvariable) ncdataset`

- type(ncvariable) `which`
- type(ncvariable) `contains`
- type(ncvariable), allocatable `the`
- type(ncvariable) `variable`
- logical `avg` = .false.
- logical `average`
- logical, dimension(:), allocatable, pointer `data`
- logical `before`
- logical `writing`
- logical, dimension(:, :), pointer `mask`
- logical, pointer `to`
- logical, pointer `reconstruct`
- real(dp), dimension(:), allocatable, pointer `data`
- real(dp), allocatable `store`
- real(dp), allocatable `the`
- real(dp), allocatable `between`
- real(dp), allocatable `writes`
- integer(i4) `counter` = 0
- integer(i4) `count`
- integer(i4), allocatable `the`
- integer(i4) `number`
- integer(i4) `of`
- integer(i4), public `updatevariable`
- integer(i4) `calls`

16.83.1 Member Function/Subroutine Documentation

16.83.1.1 `updatevariable()`

```
procedure, public mo_mrm_write_fluxes_states::outputvariable::updatevariable ( )
```

16.83.1.2 `writevariabletimestep()`

```
procedure, public mo_mrm_write_fluxes_states::outputvariable::writevariabletimestep ( )
```

16.83.2 Member Data Documentation

16.83.2.1 `average`

```
logical mo_mrm_write_fluxes_states::outputvariable::average
```

16.83.2.2 `avg`

```
logical mo_mrm_write_fluxes_states::outputvariable::avg = .false.
```

16.83.2.3 before

```
logical mo_mrm_write_fluxes_states::outputvariable::before
```

16.83.2.4 between

```
real(dp), allocatable mo_mrm_write_fluxes_states::outputvariable::between
```

16.83.2.5 calls

```
integer(i4) mo_mrm_write_fluxes_states::outputvariable::calls
```

16.83.2.6 contains

```
type(ncvariable) mo_mrm_write_fluxes_states::outputvariable::contains
```

16.83.2.7 count

```
integer(i4) mo_mrm_write_fluxes_states::outputvariable::count
```

16.83.2.8 counter

```
integer(i4) mo_mrm_write_fluxes_states::outputvariable::counter = 0
```

16.83.2.9 data [1/2]

```
logical, dimension(:), allocatable, pointer mo_mrm_write_fluxes_states::outputvariable::data
```

16.83.2.10 data [2/2]

```
real(dp), dimension(:), allocatable, pointer mo_mrm_write_fluxes_states::outputvariable::data
```

16.83.2.11 mask

```
logical, dimension(:, :), pointer mo_mrm_write_fluxes_states::outputvariable::mask
```

16.83.2.12 nc

```
type(ncvariable) mo_mrm_write_fluxes_states::outputvariable::nc
```

16.83.2.13 ncdataset

```
type(ncvariable) mo_mrm_write_fluxes_states::outputvariable::ncdataset
```

16.83.2.14 number

```
integer(i4) mo_mrm_write_fluxes_states::outputvariable::number
```

16.83.2.15 of

```
integer(i4) mo_mrm_write_fluxes_states::outputvariable::of
```

16.83.2.16 reconstruct

```
logical, pointer mo_mrm_write_fluxes_states::outputvariable::reconstruct
```

16.83.2.17 store

```
real(dp), allocatable mo_mrm_write_fluxes_states::outputvariable::store
```

16.83.2.18 the [1/3]

```
type(ncvariable), allocatable mo_mrm_write_fluxes_states::outputvariable::the
```

16.83.2.19 the [2/3]

```
real(dp), allocatable mo_mrm_write_fluxes_states::outputvariable::the
```

16.83.2.20 the [3/3]

```
integer(i4), allocatable mo_mrm_write_fluxes_states::outputvariable::the
```

16.83.2.21 to

```
logical, pointer mo_mrm_write_fluxes_states::outputvariable::to
```

16.83.2.22 updatevariable

```
integer(i4), public mo_mrm_write_fluxes_states::outputvariable::updatevariable
```

16.83.2.23 variable

```
type(ncvariable) mo_mrm_write_fluxes_states::outputvariable::variable
```

16.83.2.24 which

```
type(ncvariable) mo_mrm_write_fluxes_states::outputvariable::which
```

16.83.2.25 writes

```
real(dp), allocatable mo_mrm_write_fluxes_states::outputvariable::writes
```

16.83.2.26 writing

```
logical mo_mrm_write_fluxes_states::outputvariable::writing
```

The documentation for this interface was generated from the following file:

- [mo_mrm_write_fluxes_states.f90](#)

16.84 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)

16.84.1 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	<i>input2</i>	values to paste. Can be INTEGER(I4/I8), REAL(SP/DP), or CHARACTER(len=*) and also scalar, DIMENSION(:), or DIMENSION(:, :) If not scalar then the rows have to agree with input1
in, out	<i>allocatable :: input1</i>	array to be pasted to. Can be INTEGER(I4/I8), REAL(SP/DP), or CHARACTER(len=*). Must be DIMENSION(:) or DIMENSION(:, :), and allocatable. If input2 is not scalar then it must be size(input1,1) = size(input2,1).

Author

Juliane Mai

Date

Aug 2012

16.84.2 Member Function/Subroutine Documentation

16.84.2.1 `paste_char_m_m()`

```
subroutine mo_append::paste::paste_char_m_m (
    character(len = *), dimension(:, :), intent(inout), allocatable mat1,
    character(len = *), dimension(:, :), intent(in) mat2,
    character(len = *), intent(in), optional fill_value )
```

16.84.2.2 `paste_char_m_s()`

```
subroutine mo_append::paste::paste_char_m_s (
    character(len = *), dimension(:, :), intent(inout), allocatable mat1,
    character(len = *), intent(in) sca2 )
```

16.84.2.3 `paste_char_m_v()`

```
subroutine mo_append::paste::paste_char_m_v (
    character(len = *), dimension(:, :, ), intent(inout), allocatable mat1,
    character(len = *), dimension(:, ), intent(in) vec2,
    character(len = *), intent(in), optional fill_value )
```

16.84.2.4 `paste_dp_m_m()`

```
subroutine mo_append::paste::paste_dp_m_m (
    real(dp), dimension(:, :, ), intent(inout), allocatable mat1,
    real(dp), dimension(:, :, ), intent(in) mat2,
    real(dp), intent(in), optional fill_value )
```

16.84.2.5 `paste_dp_m_s()`

```
subroutine mo_append::paste::paste_dp_m_s (
    real(dp), dimension(:, :, ), intent(inout), allocatable mat1,
    real(dp), intent(in) sca2 )
```

16.84.2.6 `paste_dp_m_v()`

```
subroutine mo_append::paste::paste_dp_m_v (
    real(dp), dimension(:, :, ), intent(inout), allocatable mat1,
    real(dp), dimension(:, ), intent(in) vec2,
    real(dp), intent(in), optional fill_value )
```

16.84.2.7 `paste_i4_m_m()`

```
subroutine mo_append::paste::paste_i4_m_m (
    integer(i4), dimension(:, :, ), intent(inout), allocatable mat1,
    integer(i4), dimension(:, :, ), intent(in) mat2,
    integer(i4), intent(in), optional fill_value )
```

16.84.2.8 `paste_i4_m_s()`

```
subroutine mo_append::paste::paste_i4_m_s (
    integer(i4), dimension(:, :, ), intent(inout), allocatable mat1,
    integer(i4), intent(in) sca2 )
```

16.84.2.9 paste_i4_m_v()

```
subroutine mo_append::paste::paste_i4_m_v (
    integer(i4), dimension(:, :, ), intent(inout), allocatable mat1,
    integer(i4), dimension(:, ), intent(in) vec2,
    integer(i4), intent(in), optional fill_value )
```

16.84.2.10 paste_i8_m_m()

```
subroutine mo_append::paste::paste_i8_m_m (
    integer(i8), dimension(:, :, ), intent(inout), allocatable mat1,
    integer(i8), dimension(:, :, ), intent(in) mat2,
    integer(i8), intent(in), optional fill_value )
```

16.84.2.11 paste_i8_m_s()

```
subroutine mo_append::paste::paste_i8_m_s (
    integer(i8), dimension(:, :, ), intent(inout), allocatable mat1,
    integer(i8), intent(in) sca2 )
```

16.84.2.12 paste_i8_m_v()

```
subroutine mo_append::paste::paste_i8_m_v (
    integer(i8), dimension(:, :, ), intent(inout), allocatable mat1,
    integer(i8), dimension(:, ), intent(in) vec2,
    integer(i8), intent(in), optional fill_value )
```

16.84.2.13 paste_lgt_m_m()

```
subroutine mo_append::paste::paste_lgt_m_m (
    logical, dimension(:, :, ), intent(inout), allocatable mat1,
    logical, dimension(:, :, ), intent(in) mat2 )
```

16.84.2.14 paste_lgt_m_s()

```
subroutine mo_append::paste::paste_lgt_m_s (
    logical, dimension(:, :, ), intent(inout), allocatable mat1,
    logical, intent(in) sca2 )
```

16.84.2.15 paste_lgt_m_v()

```
subroutine mo_append::paste::paste_lgt_m_v (
    logical, dimension(:, :, ), intent(inout), allocatable mat1,
    logical, dimension(:, ), intent(in) vec2 )
```

16.84.2.16 `paste_sp_m_m()`

```
subroutine mo_append::paste::paste_sp_m_m (
    real(sp), dimension(:, :, ), intent(inout), allocatable mat1,
    real(sp), dimension(:, :, ), intent(in) mat2,
    real(sp), intent(in), optional fill_value )
```

16.84.2.17 `paste_sp_m_s()`

```
subroutine mo_append::paste::paste_sp_m_s (
    real(sp), dimension(:, :, ), intent(inout), allocatable mat1,
    real(sp), intent(in) sca2 )
```

16.84.2.18 `paste_sp_m_v()`

```
subroutine mo_append::paste::paste_sp_m_v (
    real(sp), dimension(:, :, ), intent(inout), allocatable mat1,
    real(sp), dimension(:, ), intent(in) vec2,
    real(sp), intent(in), optional fill_value )
```

The documentation for this interface was generated from the following file:

- [mo_append.f90](#)

16.85 `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)

16.85.1 Detailed Description

Calculates pattern dissimilarity (PD) measure.

$PD = 1 - \text{sum}(\text{dissimilarity}(\text{mat1}, \text{mat2})) / \text{count}(\text{mask})$ dissimilarity($\text{mat1}, \text{mat2}$) = 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 mismatching.

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, FTT TFT FFF |
| FFF TFT TTF, FFF FFF FFF, TTF TFT FFF |

xor=neq = | FFF FFT FFT, FFT FFT FTT, FTT FFT FFF |
| FFF TFT TTT, TTT TFT TTT, TFT TFT FFF |
| FFF TFT TTF, TTF TFT TTF, TFT TFT 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 optional 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	<i>real(sp/dp), dimension(:, :) :: mat1</i>	2D-array with input numbers
in	<i>real(sp/dp), dimension(:, :) :: mat2</i>	2D-array with input numbers
in	<i>logical, dimension(:, :), optional :: mask</i>	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	<i>logical, optional :: valid</i>	indicates if the function could determine a valid value result can be invalid 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

16.85.2 Member Function/Subroutine Documentation

16.85.2.1 pd_dp()

```
real(dp) function mo_spatialsimilarity::pd::pd_dp (
    real(dp), dimension(:, :, ), intent(in) mat1,
    real(dp), dimension(:, :, ), intent(in) mat2,
    logical, dimension(:, :, ), intent(in), optional mask,
    logical, intent(out), optional valid )
```

16.85.2.2 pd_sp()

```
real(sp) function mo_spatialsimilarity::pd::pd_sp (
    real(sp), dimension(:, :, ), intent(in) mat1,
    real(sp), dimension(:, :, ), intent(in) mat2,
    logical, dimension(:, :, ), intent(in), optional mask,
    logical, intent(out), optional valid )
```

The documentation for this interface was generated from the following file:

- [mo_spatialsimilarity.f90](#)

16.86 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)

16.86.1 Member Function/Subroutine Documentation

16.86.1.1 percentile_0d_dp()

```
real(dp) function mo_percentile::percentile::percentile_0d_dp (
    real(dp), dimension(:, ), intent(in) arrin,
    real(dp), intent(in) k,
    logical, dimension(:, ), intent(in), optional mask,
    integer(i4), intent(in), optional mode_in )
```

16.86.1.2 percentile_0d_sp()

```
real(sp) function mo_percentile::percentile::percentile_0d_sp (
    real(sp), dimension(:, ), intent(in) arrin,
    real(sp), intent(in) k,
    logical, dimension(:, ), intent(in), optional mask,
    integer(i4), intent(in), optional mode_in )
```

16.86.1.3 percentile_1d_dp()

```
real(dp) function, dimension(size(k)) mo_percentile::percentile::percentile_1d_dp (
    real(dp), dimension(:), intent(in) arrin,
    real(dp), dimension(:), intent(in) k,
    logical, dimension(:), intent(in), optional mask,
    integer(i4), intent(in), optional mode_in )
```

16.86.1.4 percentile_1d_sp()

```
real(sp) function, dimension(size(k)) mo_percentile::percentile::percentile_1d_sp (
    real(sp), dimension(:), intent(in) arrin,
    real(sp), dimension(:), intent(in) k,
    logical, dimension(:), intent(in), optional mask,
    integer(i4), intent(in), optional mode_in )
```

The documentation for this interface was generated from the following file:

- [mo_percentile.f90](#)

16.87 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**

16.87.1 Member Data Documentation

16.87.1.1 **dend**

```
integer(i4) mo_common_variables::period::dend
```

16.87.1.2 **dstart**

```
integer(i4) mo_common_variables::period::dstart
```

16.87.1.3 **julend**

```
integer(i4) mo_common_variables::period::julend
```

16.87.1.4 **julstart**

```
integer(i4) mo_common_variables::period::julstart
```

16.87.1.5 **mend**

```
integer(i4) mo_common_variables::period::mend
```

16.87.1.6 **mstart**

```
integer(i4) mo_common_variables::period::mstart
```

16.87.1.7 **nobs**

```
integer(i4) mo_common_variables::period::nobs
```

16.87.1.8 yend

```
integer(i4) mo_common_variables::period::yend
```

16.87.1.9 ystart

```
integer(i4) mo_common_variables::period::ystart
```

The documentation for this type was generated from the following file:

- [mo_common_variables.f90](#)

16.88 mo_percentile::qmedian Interface Reference

Public Member Functions

- real(sp) function [qmedian_sp](#) (dat)
- real(dp) function [qmedian_dp](#) (dat)

16.88.1 Member Function/Subroutine Documentation

16.88.1.1 qmedian_dp()

```
real(dp) function mo_percentile::qmedian::qmedian_dp (  
    real(dp), dimension(:), intent(inout) dat )
```

16.88.1.2 qmedian_sp()

```
real(sp) function mo_percentile::qmedian::qmedian_sp (   
    real(sp), dimension(:), intent(inout) dat )
```

The documentation for this interface was generated from the following file:

- [mo_percentile.f90](#)

16.89 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)

16.89.1 Member Function/Subroutine Documentation

16.89.1.1 d_rapknr()

```
subroutine mo_orderpack::rapknr::d_rapknr (
    real(kind = dp), dimension (:), intent(in) XDONT,
    integer(kind = i4), dimension (:), intent(out) IRNGT,
    integer(kind = i4), intent(in) NORD )
```

16.89.1.2 i_rapknr()

```
subroutine mo_orderpack::rapknr::i_rapknr (
    integer(kind = i4), dimension (:), intent(in) XDONT,
    integer(kind = i4), dimension (:), intent(out) IRNGT,
    integer(kind = i4), intent(in) NORD )
```

16.89.1.3 r_rapknr()

```
subroutine mo_orderpack::rapknr::r_rapknr (
    real(kind = sp), dimension (:), intent(in) XDONT,
    integer(kind = i4), dimension (:), intent(out) IRNGT,
    integer(kind = i4), intent(in) NORD )
```

The documentation for this interface was generated from the following file:

- [mo_orderpack.f90](#)

16.90 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.

16.90.1 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

16.90.2 Member Function/Subroutine Documentation

16.90.2.1 read_spatial_data_ascii_dp()

```
subroutine mo_read_spatial_data::read_spatial_data_ascii::read_spatial_data_ascii_dp (
    character(len = *), intent(in) filename,
    integer(i4), intent(in) fileunit,
    integer(i4), intent(in) header_ncols,
    integer(i4), intent(in) header_nrows,
    real(dp), intent(in) header_xllcorner,
    real(dp), intent(in) header_yllcorner,
    real(dp), intent(in) header_cellsize,
    real(dp), dimension(:, :), intent(out), allocatable data,
    logical, dimension(:, :), intent(out), allocatable mask )
```

TODO: add description.

TODO: add description

Parameters

in	character(len = *) :: filename	filename with location
in	integer(i4) :: fileunit	unit for opening the file
in	integer(i4) :: header_ncols	number of columns of data fields:
in	integer(i4) :: header_nRows	number of rows of data fields:
in	real(dp) :: header_xllcorner	header read in lower left corner
in	real(dp) :: header_yllcorner	header read in lower left corner
in	real(dp) :: header_cellsize	header read in cellsize
out	real(dp), dimension(:, :) :: data	data
out	logical, dimension(:, :) :: mask	mask

Authors

Robert Schwegelpe

Date

Jun 2018

16.90.2.2 read_spatial_data_ascii_i4()

```
subroutine mo_read_spatial_data::read_spatial_data_ascii::read_spatial_data_ascii_i4 (
    character(len = *), intent(in) filename,
    integer(i4), intent(in) fileunit,
    integer(i4), intent(in) header_ncols,
    integer(i4), intent(in) header_nrows,
    real(dp), intent(in) header_xllcorner,
    real(dp), intent(in) header_yllcorner,
    real(dp), intent(in) header_cellsize,
    integer(i4), dimension(:, :), intent(out), allocatable data,
    logical, dimension(:, :), intent(out), allocatable mask )
```

TODO: add description.

TODO: add description

Parameters

in	<i>character(len = *) :: filename</i>	filename with location
in	<i>integer(i4) :: fileunit</i>	unit for opening the file
in	<i>integer(i4) :: header_nCols</i>	number of columns of data fields:
in	<i>integer(i4) :: header_nRows</i>	number of rows of data fields:
in	<i>real(dp) :: header_xllcorner</i>	header read in lower left corner
in	<i>real(dp) :: header_yllcorner</i>	header read in lower left corner
in	<i>real(dp) :: header_cellsize</i>	header read in cellsize
out	<i>integer(i4), dimension(:, :) :: data</i>	data
out	<i>logical, dimension(:, :) :: mask</i>	mask

Authors

Robert Scheweppe

Date

Jun 2018

The documentation for this interface was generated from the following file:

- [mo_read_spatial_data.f90](#)

16.91 mo_corr::realft Interface Reference

Private Member Functions

- subroutine [realft_sp](#) (data, isign, zdata)
- subroutine [realft_dp](#) (data, isign, zdata)

16.91.1 Member Function/Subroutine Documentation

16.91.1.1 realft_dp()

```
subroutine mo_corr::realft::realft_dp (
    real(dp), dimension(:, ), intent(inout) data,
    integer(i4), intent(in) isign,
    complex(dpc), dimension(:, ), optional, target zdata ) [private]
```

16.91.1.2 realft_sp()

```
subroutine mo_corr::realft::realft_sp (
    real(sp), dimension(:), intent(inout) data,
    integer(i4), intent(in) isign,
    complex(spc), dimension(:), optional, target zdata ) [private]
```

The documentation for this interface was generated from the following file:

- [mo_corr.f90](#)

16.92 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)

16.92.1 Member Function/Subroutine Documentation

16.92.1.1 d_refpar()

```
subroutine mo_orderpack::refpar::d_refpar (
    real(kind = dp), dimension (:), intent(in) XDONT,
    integer(kind = i4), dimension (:), intent(out) IRNGT,
    integer(kind = i4), intent(in) NORD )
```

16.92.1.2 i_refpar()

```
subroutine mo_orderpack::refpar::i_refpar (
    integer(kind = i4), dimension (:), intent(in) XDONT,
    integer(kind = i4), dimension (:), intent(out) IRNGT,
    integer(kind = i4), intent(in) NORD )
```

16.92.1.3 r_refpar()

```
subroutine mo_orderpack::refpar::r_refpar (
    real(kind = sp), dimension (:), intent(in) XDONT,
    integer(kind = i4), dimension (:), intent(out) IRNGT,
    integer(kind = i4), intent(in) NORD )
```

The documentation for this interface was generated from the following file:

- [mo_orderpack.f90](#)

16.93 mo_orderpack::refsor Interface Reference

Public Member Functions

- subroutine [d_refsor](#) (XDONT)
- subroutine [r_refsor](#) (XDONT)
- subroutine [i_refsor](#) (XDONT)

16.93.1 Member Function/Subroutine Documentation

16.93.1.1 [d_refsor\(\)](#)

```
subroutine mo_orderpack::refsor::d_refsor (
    real(kind = dp), dimension (:), intent(inout) XDONT )
```

16.93.1.2 [i_refsor\(\)](#)

```
subroutine mo_orderpack::refsor::i_refsor (
    integer(kind = i4), dimension (:), intent(inout) XDONT )
```

16.93.1.3 [r_refsor\(\)](#)

```
subroutine mo_orderpack::refsor::r_refsor (
    real(kind = sp), dimension (:), intent(inout) XDONT )
```

The documentation for this interface was generated from the following file:

- [mo_orderpack.f90](#)

16.94 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)

16.94.1 Member Function/Subroutine Documentation

16.94.1.1 [d_rinpar\(\)](#)

```
subroutine mo_orderpack::rinpar::d_rinpar (
    real(kind = dp), dimension (:), intent(in) XDONT,
```

```
integer(kind = i4), dimension (:), intent(out) IRNGT,
integer(kind = i4), intent(in) NORD )
```

16.94.1.2 i_rinpar()

```
subroutine mo_orderpack::rinpar::i_rinpar (
    integer(kind = i4), dimension (:), intent(in) XDONT,
    integer(kind = i4), dimension (:), intent(out) IRNGT,
    integer(kind = i4), intent(in) NORD )
```

16.94.1.3 r_rinpar()

```
subroutine mo_orderpack::rinpar::r_rinpar (
    real(kind = sp), dimension (:), intent(in) XDONT,
    integer(kind = i4), dimension (:), intent(out) IRNGT,
    integer(kind = i4), intent(in) NORD )
```

The documentation for this interface was generated from the following file:

- [mo_orderpack.f90](#)

16.95 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)

16.95.1 Member Function/Subroutine Documentation

16.95.1.1 rmse_dp_1d()

```
real(dp) function mo_errormeasures::rmse::rmse_dp_1d (
    real(dp), dimension(:), intent(in) x,
    real(dp), dimension(:), intent(in) y,
    logical, dimension(:), intent(in), optional mask )
```

16.95.1.2 rmse_dp_2d()

```
real(dp) function mo_errormeasures::rmse::rmse_dp_2d (
    real(dp), dimension(:, :), intent(in) x,
```

```
real(dp), dimension(:, :, :), intent(in) y,
logical, dimension(:, :, :), intent(in), optional mask )
```

16.95.1.3 rmse_dp_3d()

```
real(dp) function mo_errormeasures::rmse::rmse_dp_3d (
    real(dp), dimension(:, :, :, :), intent(in) x,
    real(dp), dimension(:, :, :, :), intent(in) y,
    logical, dimension(:, :, :, :), intent(in), optional mask )
```

16.95.1.4 rmse_sp_1d()

```
real(sp) function mo_errormeasures::rmse::rmse_sp_1d (
    real(sp), dimension(:, :), intent(in) x,
    real(sp), dimension(:, :), intent(in) y,
    logical, dimension(:, :), intent(in), optional mask )
```

16.95.1.5 rmse_sp_2d()

```
real(sp) function mo_errormeasures::rmse::rmse_sp_2d (
    real(sp), dimension(:, :, :), intent(in) x,
    real(sp), dimension(:, :, :), intent(in) y,
    logical, dimension(:, :, :), intent(in), optional mask )
```

16.95.1.6 rmse_sp_3d()

```
real(sp) function mo_errormeasures::rmse::rmse_sp_3d (
    real(sp), dimension(:, :, :, :), intent(in) x,
    real(sp), dimension(:, :, :, :), intent(in) y,
    logical, dimension(:, :, :, :), intent(in), optional mask )
```

The documentation for this interface was generated from the following file:

- [mo_errormeasures.f90](#)

16.96 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)

16.96.1 Member Function/Subroutine Documentation

16.96.1.1 d_rnkpar()

```
subroutine mo_orderpack::rnkpar::d_rnkpar (
    real(kind = dp), dimension (:), intent(in) XDONT,
    integer(kind = i4), dimension (:), intent(out) IRNGT,
    integer(kind = i4), intent(in) NORD )
```

16.96.1.2 i_rnkpar()

```
subroutine mo_orderpack::rnkpar::i_rnkpar (
    integer(kind = i4), dimension (:), intent(in) XDONT,
    integer(kind = i4), dimension (:), intent(out) IRNGT,
    integer(kind = i4), intent(in) NORD )
```

16.96.1.3 r_rnkpar()

```
subroutine mo_orderpack::rnkpar::r_rnkpar (
    real(kind = sp), dimension (:), intent(in) XDONT,
    integer(kind = i4), dimension (:), intent(out) IRNGT,
    integer(kind = i4), intent(in) NORD )
```

The documentation for this interface was generated from the following file:

- [mo_orderpack.f90](#)

16.97 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)

16.97.1 Member Function/Subroutine Documentation

16.97.1.1 sae_dp_1d()

```
real(dp) function mo_errormeasures::sae::sae_dp_1d (
    real(dp), dimension(:), intent(in) x,
    real(dp), dimension(:), intent(in) y,
    logical, dimension(:), intent(in), optional mask )
```

16.97.1.2 `sae_dp_2d()`

```
real(dp) function mo_errormeasures::sae::sae_dp_2d (
    real(dp), dimension(:, :, ), intent(in) x,
    real(dp), dimension(:, :, ), intent(in) y,
    logical, dimension(:, :, ), intent(in), optional mask )
```

16.97.1.3 `sae_dp_3d()`

```
real(dp) function mo_errormeasures::sae::sae_dp_3d (
    real(dp), dimension(:, :, :, ), intent(in) x,
    real(dp), dimension(:, :, :, ), intent(in) y,
    logical, dimension(:, :, :, ), intent(in), optional mask )
```

16.97.1.4 `sae_sp_1d()`

```
real(sp) function mo_errormeasures::sae::sae_sp_1d (
    real(sp), dimension(:, ), intent(in) x,
    real(sp), dimension(:, ), intent(in) y,
    logical, dimension(:, ), intent(in), optional mask )
```

16.97.1.5 `sae_sp_2d()`

```
real(sp) function mo_errormeasures::sae::sae_sp_2d (
    real(sp), dimension(:, :, ), intent(in) x,
    real(sp), dimension(:, :, ), intent(in) y,
    logical, dimension(:, :, ), intent(in), optional mask )
```

16.97.1.6 `sae_sp_3d()`

```
real(sp) function mo_errormeasures::sae::sae_sp_3d (
    real(sp), dimension(:, :, :, ), intent(in) x,
    real(sp), dimension(:, :, :, ), intent(in) y,
    logical, dimension(:, :, :, ), intent(in), optional mask )
```

The documentation for this interface was generated from the following file:

- [mo_errormeasures.f90](#)

16.98 `mo_julian::setcalendar` Interface Reference

Private Member Functions

- subroutine `setcalendarinteger` (selector)
Set module private variable calendar.
- subroutine `setcalendarstring` (selector)
Set module private variable calendar.

16.98.1 Member Function/Subroutine Documentation

16.98.1.1 setcalendarinteger()

```
subroutine mo_julian::setcalendar::setcalendarinteger (
    integer(i4), intent(in) selector ) [private]
```

Set module private variable calendar.

Parameters

in	integer(i4) :: selector	{1 2 3}
----	-------------------------	---------

Author

Written, David Schaefer

Date

Jan 2015

16.98.1.2 setcalendarstring()

```
subroutine mo_julian::setcalendar::setcalendarstring (
    character(*), intent(in) selector ) [private]
```

Set module private variable calendar.

Parameters

in	character(len=*) :: selector	{"julian" "365day" "360day"}
----	------------------------------	------------------------------

Author

Written, David Schaefer

Date

Jan 2015

The documentation for this interface was generated from the following file:

- [mo_julian.f90](#)

16.99 mo_moment::skewness Interface Reference

Public Member Functions

- real(sp) function [skewness_sp](#) (dat, mask)
- real(dp) function [skewness_dp](#) (dat, mask)

16.99.1 Member Function/Subroutine Documentation

16.99.1.1 `skewness_dp()`

```
real(dp) function mo_moment::skewness::skewness_dp (
    real(dp), dimension(:), intent(in) dat,
    logical, dimension(:), intent(in), optional mask )
```

16.99.1.2 `skewness_sp()`

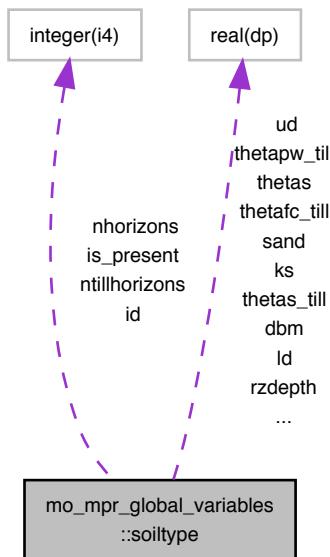
```
real(sp) function mo_moment::skewness::skewness_sp (
    real(sp), dimension(:), intent(in) dat,
    logical, dimension(:), intent(in), optional mask )
```

The documentation for this interface was generated from the following file:

- [mo_moment.f90](#)

16.100 `mo_mpr_global_variables::soiltype` Type Reference

Collaboration diagram for `mo_mpr_global_variables::soiltype`:



Private Attributes

- `integer(i4), dimension(:), allocatable id`

- integer(i4), dimension(:), allocatable `nhorizons`
- integer(i4), dimension(:), allocatable `is_present`
- real(dp), dimension(:, :), allocatable `ud`
- real(dp), dimension(:, :), allocatable `ld`
- real(dp), dimension(:, :), allocatable `clay`
- real(dp), dimension(:, :), allocatable `sand`
- real(dp), dimension(:, :), allocatable `dbm`
- real(dp), dimension(:, :), allocatable `depth`
- real(dp), dimension(:, :), allocatable `rzdepth`
- real(dp), dimension(:, :, :), allocatable `wd`
- integer(i4), dimension(:), allocatable `ntillhorizons`
- real(dp), dimension(:, :, :), allocatable `thetas_till`
- real(dp), dimension(:, :, :), allocatable `thetas`
- real(dp), dimension(:, :, :), allocatable `db`
- real(dp), dimension(:, :, :), allocatable `thetafc_till`
- real(dp), dimension(:, :, :), allocatable `thetafc`
- real(dp), dimension(:, :, :), allocatable `thetapw_till`
- real(dp), dimension(:, :, :), allocatable `thetapw`
- real(dp), dimension(:, :, :), allocatable `ks`

16.100.1 Member Data Documentation

16.100.1.1 `clay`

```
real(dp), dimension(:, :, :), allocatable mo_mpr_global_variables::soiltype::clay [private]
```

16.100.1.2 `db`

```
real(dp), dimension(:, :, :, :), allocatable mo_mpr_global_variables::soiltype::db [private]
```

16.100.1.3 `dbm`

```
real(dp), dimension(:, :, :), allocatable mo_mpr_global_variables::soiltype::dbm [private]
```

16.100.1.4 `depth`

```
real(dp), dimension(:, :, :), allocatable mo_mpr_global_variables::soiltype::depth [private]
```

16.100.1.5 `id`

```
integer(i4), dimension(:, :), allocatable mo_mpr_global_variables::soiltype::id [private]
```

16.100.1.6 is_present

```
integer(i4), dimension(:), allocatable mo_mpr_global_variables::soiltype::is_present [private]
```

16.100.1.7 ks

```
real(dp), dimension(:, :, :), allocatable mo_mpr_global_variables::soiltype::ks [private]
```

16.100.1.8 ld

```
real(dp), dimension(:, :), allocatable mo_mpr_global_variables::soiltype::ld [private]
```

16.100.1.9 nhorizons

```
integer(i4), dimension(:), allocatable mo_mpr_global_variables::soiltype::nhorizons [private]
```

16.100.1.10 ntillhorizons

```
integer(i4), dimension(:), allocatable mo_mpr_global_variables::soiltype::ntillhorizons [private]
```

16.100.1.11 rzdepth

```
real(dp), dimension(:), allocatable mo_mpr_global_variables::soiltype::rzdepth [private]
```

16.100.1.12 sand

```
real(dp), dimension(:, :), allocatable mo_mpr_global_variables::soiltype::sand [private]
```

16.100.1.13 thetafc

```
real(dp), dimension(:, :), allocatable mo_mpr_global_variables::soiltype::thetafc [private]
```

16.100.1.14 thetafc_till

```
real(dp), dimension(:, :, :), allocatable mo_mpr_global_variables::soiltype::thetafc_till [private]
```

16.100.1.15 thetapw

```
real(dp), dimension(:, :, :), allocatable mo_mpr_global_variables::soiltype::thetapw [private]
```

16.100.1.16 thetapw_till

```
real(dp), dimension(:, :, :, :), allocatable mo_mpr_global_variables::soiltype::thetapw_till [private]
```

16.100.1.17 thetas

```
real(dp), dimension(:, :, :), allocatable mo_mpr_global_variables::soiltype::thetas [private]
```

16.100.1.18 thetas_till

```
real(dp), dimension(:, :, :, :), allocatable mo_mpr_global_variables::soiltype::thetas_till [private]
```

16.100.1.19 ud

```
real(dp), dimension(:, :, :), allocatable mo_mpr_global_variables::soiltype::ud [private]
```

16.100.1.20 wd

```
real(dp), dimension(:, :, :, :), allocatable mo_mpr_global_variables::soiltype::wd [private]
```

The documentation for this type was generated from the following file:

- [mo_mpr_global_variables.f90](#)

16.101 mo_orderpack::sort Interface Reference

Unconditional ranking.

Public Member Functions

- subroutine [d_refsor](#) (XDONT)
- subroutine [r_refsor](#) (XDONT)
- subroutine [i_refsor](#) (XDONT)

16.101.1 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 REFFPAR (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 (initially inverse sorted) can easily happen. In many cases, the refined quicksort method is faster.

Integer Function INDNTH (XVALT, NORD) Returns the index of the NORDth value of XVALT (in increasing order) This routine uses a pivoting strategy such as the one of finding the median based on the quicksort algorithm, but we skew the pivot choice to try to bring it to NORD as fast as possible. It uses 2 temporary arrays, one where it stores the indices of the values smaller than the pivot, and the other for the indices of values larger than the pivot that we might still need later on. It iterates until it can bring the number of values in ILOWT to exactly NORD, and then takes out the original index of the maximum value in this set.

Subroutine INDMED (XVALT, INDM) Returns the index of the median (((Size(XVALT)+1))/2th value) of XVALT This routine uses the recursive procedure described in Knuth, The Art of Computer Programming, vol. 3, 5.3.3 - This procedure is linear in time, and does not require to be able to interpolate in the set as the one used in INDNTH. It also has better worst case behavior than INDNTH, but is about 10% slower in average for random uniformly distributed values.

Note that in Orderpack 1.0, this routine was a Function procedure, and is now changed to a Subroutine.

Unique ranking

Subroutine UNIRNK (XVALT, IRNGT, NUNI) Ranks an array, removing duplicate entries (uses merge sort). The routine is similar to pure merge-sort ranking, but on the last pass, it discards indices that correspond to duplicate entries. For performance reasons, the first 2 passes are taken out of the standard loop, and use dedicated coding.

Subroutine UNIPAR (XVALT, IRNGT, NORD) Ranks partially XVALT by IRNGT, up to order NORD at most, removing duplicate entries This routine uses a pivoting strategy such as the one of finding the median based on the quicksort algorithm, but we skew the pivot choice to try to bring it to NORD as quickly as possible. It uses 2 temporary arrays, one where it stores the indices of the values smaller than the pivot, and the other for the indices of values larger than the pivot that we might still need later on. It iterates until it can bring the number of values in ILOWT to exactly NORD, and then uses an insertion sort to rank this set, since it is supposedly small. At all times, the NORD first values in ILOWT correspond to distinct values of the input array.

Subroutine UNIINV (XVALT, IGOEST) Inverse ranking of an array, with removal of duplicate entries The routine is similar to pure merge-sort ranking, but on the last pass, it sets indices in IGOEST to the rank of the original value in an ordered set with duplicates removed. For performance reasons, the first 2 passes are taken out of the standard loop, and use dedicated coding.

Subroutine MULCNT (XVALT, IMULT) Gives, for each array value, its multiplicity The number of times that a value appears in the array is computed by using inverse ranking, counting for each rank the number of values that "collide"

to this rank, and returning this sum to the locations in the original set. Uses subroutine UNIINV.

Random permutation: an interesting use of ranking

A variation of the following problem was raised on the internet sci.math.num-analysis news group: Given an array, I would like to find a random permutation of this array that I could control with a `nearbiness` parameter so that elements stay close to their initial locations. The `nearbiness` 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. correlation 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 (initially 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.

16.101.2 Member Function/Subroutine Documentation

16.101.2.1 d_refsor()

```
subroutine mo_orderpack::sort::d_refsor (
    real(kind = dp), dimension (:), intent(inout) XDONT )
```

16.101.2.2 i_refsor()

```
subroutine mo_orderpack::sort::i_refsor (
    integer(kind = i4), dimension (:), intent(inout) XDONT )
```

16.101.2.3 r_refsor()

```
subroutine mo_orderpack::sort::r_refsor (
    real(kind = sp), dimension (:), intent(inout) XDONT )
```

The documentation for this interface was generated from the following file:

- [mo_orderpack.f90](#)

16.102 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)

16.102.1 Member Function/Subroutine Documentation

16.102.1.1 sort_index_dp()

```
integer(i4) function, dimension(size(arr)) mo_orderpack::sort_index::sort_index_dp (
    real(dp), dimension(:), intent(in) arr )
```

16.102.1.2 sort_index_i4()

```
integer(i4) function, dimension(size(arr)) mo_orderpack::sort_index::sort_index_i4 (
    integer(i4), dimension(:), intent(in) arr )
```

16.102.1.3 sort_index_sp()

```
integer(i4) function, dimension(size(arr)) mo_orderpack::sort_index::sort_index_sp (
    real(sp), dimension(:), intent(in) arr )
```

The documentation for this interface was generated from the following file:

- [mo_orderpack.f90](#)

16.103 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)

16.103.1 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

16.103.2 Member Function/Subroutine Documentation

16.103.2.1 spatial_aggregation_3d()

```
subroutine mo_spatial_agg_disagg_forcing::spatial_aggregation::spatial_aggregation_3d (
    real(dp), dimension(:, :, :), intent(in) data2,
    real(dp), intent(in) cellsize2,
    real(dp), intent(in) cellsize1,
    logical, dimension(:, :, :), intent(in) mask1,
    logical, dimension(:, :, :), intent(in) mask2,
    real(dp), dimension(:, :, :), intent(out), allocatable data1 )
```

16.103.2.2 spatial_aggregation_4d()

```
subroutine mo_spatial_agg_disagg_forcing::spatial_aggregation::spatial_aggregation_4d (
    real(dp), dimension(:, :, :, :), intent(in) data2,
    real(dp), intent(in) cellsize2,
    real(dp), intent(in) cellsize1,
    logical, dimension(:, :, :), intent(in) mask1,
    logical, dimension(:, :, :), intent(in) mask2,
    real(dp), dimension(:, :, :, :), intent(out), allocatable data1 )
```

The documentation for this interface was generated from the following file:

- [mo_spatial_agg_disagg_forcing.f90](#)

16.104 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)

16.104.1 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	<i>real(dp), dimension(:, :, :) :: data2</i>	Level-2 data
in	<i>real(dp) :: cellsize2</i>	Level-2 resolution
in	<i>real(dp) :: cellsize1</i>	Level-1 resolution
in	<i>logical, dimension(:, :) :: mask1</i>	Level-1 mask
in	<i>logical, dimension(:, :) :: mask2</i>	Level-2 mask
out	<i>real(dp), dimension(:, :, :) :: data1</i>	Level-1 data

Authors

Rohini Kumar

Date

Jan 2013

16.104.2 Member Function/Subroutine Documentation

16.104.2.1 spatial_disaggregation_3d()

```
subroutine mo_spatial_agg_disagg_forcing::spatial_disaggregation::spatial_disaggregation_3d (
    real(dp), dimension(:, :, :), intent(in) data2,
    real(dp), intent(in) cellsize2,
    real(dp), intent(in) cellsize1,
    logical, dimension(:, :, :), intent(in) mask1,
    logical, dimension(:, :, :), intent(in) mask2,
    real(dp), dimension(:, :, :, :), intent(out), allocatable data1 )
```

16.104.2.2 spatial_disaggregation_4d()

```
subroutine mo_spatial_agg_disagg_forcing::spatial_disaggregation::spatial_disaggregation_4d (
    real(dp), dimension(:, :, :, :, :), intent(in) data2,
    real(dp), intent(in) cellsize2,
    real(dp), intent(in) cellsize1,
    logical, dimension(:, :, :), intent(in) mask1,
    logical, dimension(:, :, :), intent(in) mask2,
    real(dp), dimension(:, :, :, :, :), intent(out), allocatable data1 )
```

The documentation for this interface was generated from the following file:

- [mo_spatial_agg_disagg_forcing.f90](#)

16.105 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)

16.105.1 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	<i>real(sp/dp) :: x</i>	dummy for kind of output
in	<i>"character(len=*)</i>	:: name ieee signal name

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

16.105.2 Member Function/Subroutine Documentation

16.105.2.1 special_value_dp()

```
real(dp) function mo_utils::special_value::special_value_dp (
    real(dp), intent(in) x,
    character(len = *), intent(in) ieee )
```

16.105.2.2 special_value_sp()

```
real(sp) function mo_utils::special_value::special_value_sp (
    real(sp), intent(in) x,
    character(len = *), intent(in) ieee )
```

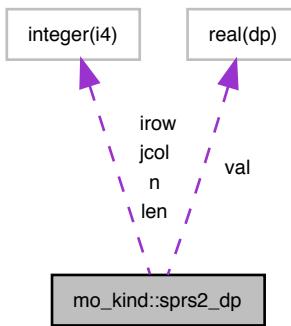
The documentation for this interface was generated from the following file:

- [mo_utils.f90](#)

16.106 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`

16.106.1 Detailed Description

Double Precision Numerical Recipes types for sparse arrays.

16.106.2 Member Data Documentation

16.106.2.1 irow

```
integer(i4), dimension(:), pointer mo_kind::sprs2_dp::irow
```

16.106.2.2 jcol

```
integer(i4), dimension(:), pointer mo_kind::sprs2_dp::jcol
```

16.106.2.3 len

```
integer(i4) mo_kind::sprs2_dp::len
```

16.106.2.4 n

```
integer(i4) mo_kind::sprs2_dp::n
```

16.106.2.5 val

```
real(dp), dimension(:), pointer mo_kind::sprs2_dp::val
```

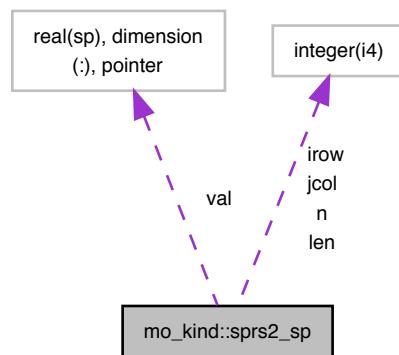
The documentation for this type was generated from the following file:

- [mo_kind.f90](#)

16.107 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`

16.107.1 Detailed Description

Single Precision Numerical Recipes types for sparse arrays.

16.107.2 Member Data Documentation

16.107.2.1 `irow`

```
integer(i4), dimension(:), pointer mo_kind::sprs2_sp::irow
```

16.107.2.2 `jcol`

```
integer(i4), dimension(:), pointer mo_kind::sprs2_sp::jcol
```

16.107.2.3 `len`

```
integer(i4) mo_kind::sprs2_sp::len
```

16.107.2.4 `n`

```
integer(i4) mo_kind::sprs2_sp::n
```

16.107.2.5 `val`

```
real(sp), dimension(:), pointer mo_kind::sprs2_sp::val
```

The documentation for this type was generated from the following file:

- [mo_kind.f90](#)

16.108 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`)

16.108.1 Member Function/Subroutine Documentation

16.108.1.1 sse_dp_1d()

```
real(dp) function mo_errormeasures::sse::sse_dp_1d (
    real(dp), dimension(:), intent(in) x,
    real(dp), dimension(:), intent(in) y,
    logical, dimension(:), intent(in), optional mask )
```

16.108.1.2 sse_dp_2d()

```
real(dp) function mo_errormeasures::sse::sse_dp_2d (
    real(dp), dimension(:, :), intent(in) x,
    real(dp), dimension(:, :), intent(in) y,
    logical, dimension(:, :), intent(in), optional mask )
```

16.108.1.3 sse_dp_3d()

```
real(dp) function mo_errormeasures::sse::sse_dp_3d (
    real(dp), dimension(:, :, :), intent(in) x,
    real(dp), dimension(:, :, :), intent(in) y,
    logical, dimension(:, :, :), intent(in), optional mask )
```

16.108.1.4 sse_sp_1d()

```
real(sp) function mo_errormeasures::sse::sse_sp_1d (
    real(sp), dimension(:), intent(in) x,
    real(sp), dimension(:), intent(in) y,
    logical, dimension(:), intent(in), optional mask )
```

16.108.1.5 sse_sp_2d()

```
real(sp) function mo_errormeasures::sse::sse_sp_2d (
    real(sp), dimension(:, :), intent(in) x,
    real(sp), dimension(:, :), intent(in) y,
    logical, dimension(:, :), intent(in), optional mask )
```

16.108.1.6 sse_sp_3d()

```
real(sp) function mo_errormeasures::sse::sse_sp_3d (
    real(sp), dimension(:, :, :), intent(in) x,
    real(sp), dimension(:, :, :), intent(in) y,
    logical, dimension(:, :, :), intent(in), optional mask )
```

The documentation for this interface was generated from the following file:

- [mo_errormeasures.f90](#)

16.109 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)`

16.109.1 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).

$$\text{standard_score} = \frac{x - \mu_x}{\sigma_x}$$

where μ_x is the mean of a population x and σ_x its standard deviation.

If an optional 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

<code>in</code>	<code>real(sp/dp), dimension(:) :: data</code>	data to calculate the standard score for
<code>in</code>	<code>logical, dimension(:),optional :: mask</code>	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

16.109.2 Member Function/Subroutine Documentation

16.109.2.1 standard_score_dp()

```
real(dp) function, dimension(size(data, dim = 1)) mo_standard_score::standard_score_dp (
    real(dp), dimension(:), intent(in) data,
    logical, dimension(:), intent(in), optional mask )
```

16.109.2.2 standard_score_sp()

```
real(sp) function, dimension(size(data, dim = 1)) mo_standard_score::standard_score_sp (
    real(sp), dimension(:), intent(in) data,
    logical, dimension(:), intent(in), optional mask )
```

The documentation for this interface was generated from the following file:

- [mo_standard_score.f90](#)

16.110 mo_moment::stddev Interface Reference

Public Member Functions

- `real(sp) function stddev_sp (dat, mask)`
- `real(dp) function stddev_dp (dat, mask)`

16.110.1 Member Function/Subroutine Documentation

16.110.1.1 stddev_dp()

```
real(dp) function mo_moment::stddev::stddev_dp (
    real(dp), dimension(:), intent(in) dat,
    logical, dimension(:), intent(in), optional mask )
```

16.110.1.2 stddev_sp()

```
real(sp) function mo_moment::stddev::stddev_sp (
    real(sp), dimension(:), intent(in) dat,
    logical, dimension(:), intent(in), optional mask )
```

The documentation for this interface was generated from the following file:

- [mo_moment.f90](#)

16.111 mo_corr::swap Interface Reference

Private Member Functions

- subroutine [swap_1d_spc](#) (a, b)
- subroutine [swap_1d_dpc](#) (a, b)

16.111.1 Member Function/Subroutine Documentation

16.111.1.1 [swap_1d_dpc\(\)](#)

```
subroutine mo_corr::swap::swap_1d_dpc (
    complex(dpc), dimension(:), intent(inout) a,
    complex(dpc), dimension(:), intent(inout) b )  [private]
```

16.111.1.2 [swap_1d_spc\(\)](#)

```
subroutine mo_corr::swap::swap_1d_spc (
    complex(spc), dimension(:), intent(inout) a,
    complex(spc), dimension(:), intent(inout) b )  [private]
```

The documentation for this interface was generated from the following file:

- [mo_corr.f90](#)

16.112 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)

16.112.1 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	<i>integer(i4) :: i</i>	Index of first element to be swapped with second [case swap(vec,i,j)]
----	-------------------------	---

Parameters

in	<i>integer(i4) :: j</i>	Index of second element to be swapped with first [case swap(vec,i,j)]
in,out	<i>real(dp/dp/i4) :: x[(:,...)]</i>	First scalar or array to swap with second [case swap(x,y)]
in,out	<i>real(dp/dp/i4) :: y[(:,::)]</i>	Second scalar or array to swap with first [case swap(x,y)]
in,out	<i>real(dp/dp/i4) :: x(:)</i>	Vector of which to elements are swapped [case swap(vec,i,j)]

Author

Matthias Cuntz

Date

May 2014

16.112.2 Member Function/Subroutine Documentation

16.112.2.1 swap_vec_dp()

```
subroutine mo_utils::swap::swap_vec_dp (
    real(dp), dimension(:), intent(inout) x,
    integer(i4), intent(in) i1,
    integer(i4), intent(in) i2 )
```

16.112.2.2 swap_vec_i4()

```
subroutine mo_utils::swap::swap_vec_i4 (
    integer(i4), dimension(:), intent(inout) x,
    integer(i4), intent(in) i1,
    integer(i4), intent(in) i2 )
```

16.112.2.3 swap_vec_sp()

```
subroutine mo_utils::swap::swap_vec_sp (
    real(sp), dimension(:), intent(inout) x,
    integer(i4), intent(in) i1,
    integer(i4), intent(in) i2 )
```

16.112.2.4 swap_xy_dp()

```
elemental pure subroutine mo_utils::swap::swap_xy_dp (
    real(dp), intent(inout) x,
    real(dp), intent(inout) y )
```

16.112.2.5 swap_xy_i4()

```
elemental pure subroutine mo_utils::swap::swap_xy_i4 (
    integer(i4), intent(inout) x,
    integer(i4), intent(inout) y )
```

16.112.2.6 swap_xy_sp()

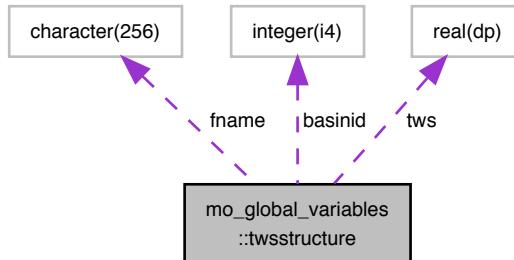
```
elemental pure subroutine mo_utils::swap::swap_xy_sp (
    real(sp), intent(inout) x,
    real(sp), intent(inout) y )
```

The documentation for this interface was generated from the following file:

- [mo_utils.f90](#)

16.113 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](#)

16.113.1 Member Data Documentation

16.113.1.1 basinid

```
integer(i4), dimension(:), allocatable mo_global_variables::twsstructure::basinid
```

16.113.1.2 fname

```
character(256), dimension(:), allocatable mo_global_variables::twsstructure::fname
```

16.113.1.3 tws

```
real(dp), dimension(:, :), allocatable mo_global_variables::twsstructure::tws
```

The documentation for this type was generated from the following file:

- [mo_global_variables.f90](#)

16.114 mo_orderpack::uniinv Interface Reference

Public Member Functions

- subroutine [d_uniinv](#) (XDONT, IGOEST)
- subroutine [r_uniinv](#) (XDONT, IGOEST)
- subroutine [i_uniinv](#) (XDONT, IGOEST)

16.114.1 Member Function/Subroutine Documentation

16.114.1.1 d_uniinv()

```
subroutine mo_orderpack::uniinv::d_uniinv (
    real(kind = dp), dimension (:), intent(in) XDONT,
    integer(kind = i4), dimension (:), intent(out) IGOEST )
```

16.114.1.2 i_uniinv()

```
subroutine mo_orderpack::uniinv::i_uniinv (
    integer(kind = i4), dimension (:), intent(in) XDONT,
    integer(kind = i4), dimension (:), intent(out) IGOEST )
```

16.114.1.3 r_uniinv()

```
subroutine mo_orderpack::uniinv::r_uniinv (
    real(kind = sp), dimension (:), intent(in) XDONT,
    integer(kind = i4), dimension (:), intent(out) IGOEST )
```

The documentation for this interface was generated from the following file:

- [mo_orderpack.f90](#)

16.115 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)

16.115.1 Member Function/Subroutine Documentation

16.115.1.1 [d_unipar\(\)](#)

```
subroutine mo_orderpack::unipar::d_unipar (
    real(kind = dp), dimension (:), intent(in) XDONT,
    integer(kind = i4), dimension (:), intent(out) IRNGT,
    integer(kind = i4), intent(inout) NORD )
```

16.115.1.2 [i_unipar\(\)](#)

```
subroutine mo_orderpack::unipar::i_unipar (
    integer(kind = i4), dimension (:), intent(in) XDONT,
    integer(kind = i4), dimension (:), intent(out) IRNGT,
    integer(kind = i4), intent(inout) NORD )
```

16.115.1.3 [r_unipar\(\)](#)

```
subroutine mo_orderpack::unipar::r_unipar (
    real(kind = sp), dimension (:), intent(in) XDONT,
    integer(kind = i4), dimension (:), intent(out) IRNGT,
    integer(kind = i4), intent(inout) NORD )
```

The documentation for this interface was generated from the following file:

- [mo_orderpack.f90](#)

16.116 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)

16.116.1 Member Function/Subroutine Documentation

16.116.1.1 d_unirnk()

```
subroutine mo_orderpack::unirnk::d_unirnk (
    real(kind = dp), dimension (:), intent(in) XVALT,
    integer(kind = i4), dimension (:), intent(out) IRNGT,
    integer(kind = i4), intent(out) NUNI )
```

16.116.1.2 i_unirnk()

```
subroutine mo_orderpack::unirnk::i_unirnk (
    integer(kind = i4), dimension (:), intent(in) XVALT,
    integer(kind = i4), dimension (:), intent(out) IRNGT,
    integer(kind = i4), intent(out) NUNI )
```

16.116.1.3 r_unirnk()

```
subroutine mo_orderpack::unirnk::r_unirnk (
    real(kind = sp), dimension (:), intent(in) XVALT,
    integer(kind = i4), dimension (:), intent(out) IRNGT,
    integer(kind = i4), intent(out) NUNI )
```

The documentation for this interface was generated from the following file:

- [mo_orderpack.f90](#)

16.117 mo_orderpack::unista Interface Reference

Public Member Functions

- subroutine [d_unista](#) (XDONT, NUNI)
- subroutine [r_unista](#) (XDONT, NUNI)
- subroutine [i_unista](#) (XDONT, NUNI)

16.117.1 Member Function/Subroutine Documentation

16.117.1.1 d_unista()

```
subroutine mo_orderpack::unista::d_unista (
    real(kind = dp), dimension (:), intent(inout) XDONT,
    integer(kind = i4), intent(out) NUNI )
```

16.117.1.2 i_unista()

```
subroutine mo_orderpack::unista::i_unista (
    integer(kind = i4), dimension (:), intent(inout) XDONT,
    integer(kind = i4), intent(out) NUNI )
```

16.117.1.3 r_unista()

```
subroutine mo_orderpack::unista::r_unista (
    real(kind = sp), dimension (:), intent(inout) XDONT,
    integer(kind = i4), intent(out) NUNI )
```

The documentation for this interface was generated from the following file:

- [mo_orderpack.f90](#)

16.118 mo_mpr_restart::unpack_field_and_write Interface Reference

TODO: add description.

Private Member Functions

- subroutine [unpack_field_and_write_1d_i4](#) (nc, var_name, var_dims, fill_value, data, mask, var_long_name)
- subroutine [unpack_field_and_write_1d_dp](#) (nc, var_name, var_dims, fill_value, data, mask, var_long_name)
- subroutine [unpack_field_and_write_2d_dp](#) (nc, var_name, var_dims, fill_value, data, mask, var_long_name)
- subroutine [unpack_field_and_write_3d_dp](#) (nc, var_name, var_dims, fill_value, data, mask, var_long_name)

16.118.1 Detailed Description

TODO: add description.

TODO: add description

Parameters

in, out	<i>type(NcDataset) :: nc</i>	NcDataset to add variable to
in	<i>character(*) :: var_name</i>	variable name
in	<i>type(NcDimension), dimension(:) :: var_dims</i>	vector of Variable dimensions
in	<i>integer(i4) :: fill_value</i>	fill value used for missing values
in	<i>integer(i4), dimension(:) :: data</i>	packed data to be set to variable
in	<i>logical, dimension(:, :) :: mask</i>	mask used for unpacking
in	<i>character(*), optional :: var_long_name</i>	variable long name attribute

Authors

Robert Scheweppe

Date

Jun 2018

16.118.2 Member Function/Subroutine Documentation

16.118.2.1 unpack_field_and_write_1d_dp()

```
subroutine mo_mpr_restart::unpack_field_and_write::unpack_field_and_write_1d_dp (
    type(ncdataset), intent(inout) nc,
```

```

character(*), intent(in) var_name,
type(ncdimension), dimension(:), intent(in) var_dims,
real(dp), intent(in) fill_value,
real(dp), dimension(:), intent(in) data,
logical, dimension(:, :, :), intent(in) mask,
character(*), intent(in), optional var_long_name ) [private]

```

16.118.2.2 unpack_field_and_write_1d_i4()

```

subroutine mo_mpr_restart::unpack_field_and_write::unpack_field_and_write_1d_i4 (
    type(ncdataset), intent(inout) nc,
    character(*), intent(in) var_name,
    type(ncdimension), dimension(:), intent(in) var_dims,
    integer(i4), intent(in) fill_value,
    integer(i4), dimension(:), intent(in) data,
    logical, dimension(:, :, :), intent(in) mask,
    character(*), intent(in), optional var_long_name ) [private]

```

16.118.2.3 unpack_field_and_write_2d_dp()

```

subroutine mo_mpr_restart::unpack_field_and_write::unpack_field_and_write_2d_dp (
    type(ncdataset), intent(inout) nc,
    character(*), intent(in) var_name,
    type(ncdimension), dimension(:), intent(in) var_dims,
    real(dp), intent(in) fill_value,
    real(dp), dimension(:, :, :), intent(in) data,
    logical, dimension(:, :, :), intent(in) mask,
    character(*), intent(in), optional var_long_name ) [private]

```

16.118.2.4 unpack_field_and_write_3d_dp()

```

subroutine mo_mpr_restart::unpack_field_and_write::unpack_field_and_write_3d_dp (
    type(ncdataset), intent(inout) nc,
    character(*), intent(in) var_name,
    type(ncdimension), dimension(:), intent(in) var_dims,
    real(dp), intent(in) fill_value,
    real(dp), dimension(:, :, :, :), intent(in) data,
    logical, dimension(:, :, :, :), intent(in) mask,
    character(*), intent(in), optional var_long_name ) [private]

```

The documentation for this interface was generated from the following file:

- [mo_mpr_restart.f90](#)

16.119 mo_restart::unpack_field_and_write Interface Reference

TODO: add description.

Private Member Functions

- subroutine [unpack_field_and_write_1d_i4](#) (nc, var_name, var_dims, fill_value, data, mask, var_long_name)
- subroutine [unpack_field_and_write_1d_dp](#) (nc, var_name, var_dims, fill_value, data, mask, var_long_name)
- subroutine [unpack_field_and_write_2d_dp](#) (nc, var_name, var_dims, fill_value, data, mask, var_long_name)
- subroutine [unpack_field_and_write_3d_dp](#) (nc, var_name, var_dims, fill_value, data, mask, var_long_name)

16.119.1 Detailed Description

TODO: add description.

TODO: add description

Parameters

in, out	<i>type(NcDataset) :: nc</i>	NcDataset to add variable to
in	<i>character(*) :: var_name</i>	variable name
in	<i>type(NcDimension), dimension(:) :: var_dims</i>	vector of Variable dimensions
in	<i>integer(i4) :: fill_value</i>	fill value used for missing values
in	<i>integer(i4), dimension(:) :: data</i>	packed data to be set to variable
in	<i>logical, dimension(:, :) :: mask</i>	mask used for unpacking
in	<i>character(*), optional :: var_long_name</i>	variable long name attribute

Authors

Robert Schwepppe

Date

Jun 2018

16.119.2 Member Function/Subroutine Documentation

16.119.2.1 [unpack_field_and_write_1d_dp\(\)](#)

```
subroutine mo_restart::unpack_field_and_write::unpack_field_and_write_1d_dp (
    type(ncdataset), intent(inout) nc,
    character(*), intent(in) var_name,
    type(ncdimension), dimension(:), intent(in) var_dims,
    real(dp), intent(in) fill_value,
    real(dp), dimension(:, :), intent(in) data,
    logical, dimension(:, :, :), intent(in) mask,
    character(*), intent(in), optional var_long_name ) [private]
```

16.119.2.2 [unpack_field_and_write_1d_i4\(\)](#)

```
subroutine mo_restart::unpack_field_and_write::unpack_field_and_write_1d_i4 (
    type(ncdataset), intent(inout) nc,
    character(*), intent(in) var_name,
```

```

type(ncdimension), dimension(:), intent(in) var_dims,
integer(i4), intent(in) fill_value,
integer(i4), dimension(:), intent(in) data,
logical, dimension(:, :, :), intent(in) mask,
character(*), intent(in), optional var_long_name ) [private]

```

16.119.2.3 unpack_field_and_write_2d_dp()

```

subroutine mo_restart::unpack_field_and_write::unpack_field_and_write_2d_dp (
    type(ncdataset), intent(inout) nc,
    character(*), intent(in) var_name,
    type(ncdimension), dimension(:), intent(in) var_dims,
    real(dp), intent(in) fill_value,
    real(dp), dimension(:, :, :), intent(in) data,
    logical, dimension(:, :, :), intent(in) mask,
    character(*), intent(in), optional var_long_name ) [private]

```

16.119.2.4 unpack_field_and_write_3d_dp()

```

subroutine mo_restart::unpack_field_and_write::unpack_field_and_write_3d_dp (
    type(ncdataset), intent(inout) nc,
    character(*), intent(in) var_name,
    type(ncdimension), dimension(:), intent(in) var_dims,
    real(dp), intent(in) fill_value,
    real(dp), dimension(:, :, :, :), intent(in) data,
    logical, dimension(:, :, :, :), intent(in) mask,
    character(*), intent(in), optional var_long_name ) [private]

```

The documentation for this interface was generated from the following file:

- [mo_restart.f90](#)

16.120 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)

16.120.1 Member Function/Subroutine Documentation

16.120.1.1 [d_valmed\(\)](#)

```

recursive real(kind = dp) function mo_orderpack::valmed::d_valmed (
    real(kind = dp), dimension (:), intent(in) XDONT )

```

16.120.1.2 i_valmed()

```
recursive integer(kind = i4) function mo_orderpack::valmed::i_valmed (
    integer(kind = i4), dimension (:), intent(in) XDONT )
```

16.120.1.3 r_valmed()

```
recursive real(kind = sp) function mo_orderpack::valmed::r_valmed (
    real(kind = sp), dimension (:), intent(in) XDONT )
```

The documentation for this interface was generated from the following file:

- [mo_orderpack.f90](#)

16.121 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)

16.121.1 Member Function/Subroutine Documentation

16.121.1.1 d_valnth()

```
real(kind = dp) function mo_orderpack::valnth::d_valnth (
    real(kind = dp), dimension (:), intent(in) XDONT,
    integer(kind = i4), intent(in) NORD )
```

16.121.1.2 i_valnth()

```
integer(kind = i4) function mo_orderpack::valnth::i_valnth (
    integer(kind = i4), dimension (:), intent(in) XDONT,
    integer(kind = i4), intent(in) NORD )
```

16.121.1.3 r_valnth()

```
real(kind = sp) function mo_orderpack::valnth::r_valnth (
    real(kind = sp), dimension (:), intent(in) XDONT,
    integer(kind = i4), intent(in) NORD )
```

The documentation for this interface was generated from the following file:

- [mo_orderpack.f90](#)

16.122 mo_ncwrite::var2nc Interface Reference

Extended [dump_ncdf](#) 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)

16.122.1 Detailed Description

Extended [dump_ncdf](#) 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	<i>character(*) :: f_name</i>	filename
in	<i>integer(i4)/real(sp,dp) :: arr(:,:,:,:)</i>	array to write
in	<i>character(*) :: dnames(:)</i>	dimension names
in	<i>character(*) :: v_name</i>	variable name
in	<i>integer(i4), optional :: dim_unlimited</i>	index of unlimited dimension
in	<i>character(*), optional :: long_name</i>	attribute
in	<i>character(*), optional :: units</i>	attribute

Parameters

in	<i>integer(i4)/real(sp,dp), optional :: missing_value</i>	attribute
in	<i>character(256), dimension(:, :), optional :: attributes</i>	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	<i>logical, optional :: create</i>	flag - specify whether a output file should be created, default
in, out	<i>integer(i4)/real(sp,dp), optional :: ncid</i>	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	<i>integer(i4), optional :: nrec</i>	if given: start point on unlimited dimension.

Note

It is not allowed to write the following 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

16.122.2 Member Function/Subroutine Documentation

16.122.2.1 var2nc_1d_dp()

```
subroutine mo_ncwrite::var2nc::var2nc_1d_dp (
    character(len = *), intent(in) f_name,
    real(dp), dimension(:, :), intent(in) arr,
    character(len = *), dimension(:, :), intent(in) dnames,
    character(len = *), intent(in) v_name,
    integer(i4), intent(in), optional dim_unlimited,
    character(len = *), intent(in), optional long_name,
    character(len = *), intent(in), optional units,
    real(dp), intent(in), optional missing_value,
    character(256), dimension(:, :, :), intent(in), optional attributes,
    logical, intent(in), optional create,
```

```
integer(i4), intent(inout), optional ncid,
integer(i4), intent(in), optional nrec )
```

16.122.2.2 var2nc_1d_i4()

```
subroutine mo_ncwrite::var2nc::var2nc_1d_i4 (
    character(len = *), intent(in) f_name,
    integer(i4), dimension(:), intent(in) arr,
    character(len = *), dimension(:), intent(in) dnames,
    character(len = *), intent(in), 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 )
```

16.122.2.3 var2nc_1d_sp()

```
subroutine mo_ncwrite::var2nc::var2nc_1d_sp (
    character(len = *), intent(in) f_name,
    real(sp), dimension(:), intent(in) arr,
    character(len = *), dimension(:), intent(in) dnames,
    character(len = *), intent(in) v_name,
    integer(i4), intent(in), optional dim_unlimited,
    character(len = *), intent(in), optional long_name,
    character(len = *), intent(in), optional units,
    real(sp), intent(in), optional missing_value,
    character(256), dimension(:, :), intent(in), optional attributes,
    logical, intent(in), optional create,
    integer(i4), intent(inout), optional ncid,
    integer(i4), intent(in), optional nrec )
```

16.122.2.4 var2nc_2d_dp()

```
subroutine mo_ncwrite::var2nc::var2nc_2d_dp (
    character(len = *), intent(in) f_name,
    real(dp), dimension(:, :), intent(in) arr,
    character(len = *), dimension(:), intent(in) dnames,
    character(len = *), intent(in) v_name,
    integer(i4), intent(in), optional dim_unlimited,
    character(len = *), intent(in), optional long_name,
    character(len = *), intent(in), optional units,
    real(dp), intent(in), optional missing_value,
    character(256), dimension(:, :), intent(in), optional attributes,
    logical, intent(in), optional create,
    integer(i4), intent(inout), optional ncid,
    integer(i4), intent(in), optional nrec )
```

16.122.2.5 var2nc_2d_i4()

```
subroutine mo_ncwrite::var2nc::var2nc_2d_i4 (
    character(len = *), intent(in) f_name,
    integer(i4), dimension(:, :, :), intent(in) arr,
    character(len = *), dimension(:, :), intent(in) dnames,
    character(len = *), intent(in) v_name,
    integer(i4), intent(in), optional dim_unlimited,
    character(len = *), intent(in), optional long_name,
    character(len = *), intent(in), optional units,
    integer(i4), intent(in), optional missing_value,
    character(256), dimension(:, :, :), intent(in), optional attributes,
    logical, intent(in), optional create,
    integer(i4), intent(inout), optional ncid,
    integer(i4), intent(in), optional nrec )
```

16.122.2.6 var2nc_2d_sp()

```
subroutine mo_ncwrite::var2nc::var2nc_2d_sp (
    character(len = *), intent(in) f_name,
    real(sp), dimension(:, :, :), intent(in) arr,
    character(len = *), dimension(:, :), intent(in) dnames,
    character(len = *), intent(in) v_name,
    integer(i4), intent(in), optional dim_unlimited,
    character(len = *), intent(in), optional long_name,
    character(len = *), intent(in), optional units,
    real(sp), intent(in), optional missing_value,
    character(256), dimension(:, :, :), intent(in), optional attributes,
    logical, intent(in), optional create,
    integer(i4), intent(inout), optional ncid,
    integer(i4), intent(in), optional nrec )
```

16.122.2.7 var2nc_3d_dp()

```
subroutine mo_ncwrite::var2nc::var2nc_3d_dp (
    character(len = *), intent(in) f_name,
    real(dp), dimension(:, :, :, :), intent(in) arr,
    character(len = *), dimension(:, :), intent(in) dnames,
    character(len = *), intent(in) v_name,
    integer(i4), intent(in), optional dim_unlimited,
    character(len = *), intent(in), optional long_name,
    character(len = *), intent(in), optional units,
    real(dp), intent(in), optional missing_value,
    character(256), dimension(:, :, :), intent(in), optional attributes,
    logical, intent(in), optional create,
    integer(i4), intent(inout), optional ncid,
    integer(i4), intent(in), optional nrec )
```

16.122.2.8 var2nc_3d_i4()

```
subroutine mo_ncwrite::var2nc::var2nc_3d_i4 (
    character(len = *), intent(in) f_name,
```

```

integer(i4), dimension(:, :, :), intent(in) arr,
character(len = *), dimension(:), intent(in) dnames,
character(len = *), intent(in) v_name,
integer(i4), intent(in), optional dim_unlimited,
character(len = *), intent(in), optional long_name,
character(len = *), intent(in), optional units,
integer(i4), intent(in), optional missing_value,
character(256), dimension(:, :, :), intent(in), optional attributes,
logical, intent(in), optional create,
integer(i4), intent(inout), optional ncid,
integer(i4), intent(in), optional nrec )

```

16.122.2.9 var2nc_3d_sp()

```

subroutine mo_ncwrite::var2nc::var2nc_3d_sp (
    character(len = *), intent(in) f_name,
    real(sp), dimension(:, :, :, :), intent(in) arr,
    character(len = *), dimension(:), intent(in) dnames,
    character(len = *), intent(in) v_name,
    integer(i4), intent(in), optional dim_unlimited,
    character(len = *), intent(in), optional long_name,
    character(len = *), intent(in), optional units,
    real(sp), intent(in), optional missing_value,
    character(256), dimension(:, :, :), intent(in), optional attributes,
    logical, intent(in), optional create,
    integer(i4), intent(inout), optional ncid,
    integer(i4), intent(in), optional nrec )

```

16.122.2.10 var2nc_4d_dp()

```

subroutine mo_ncwrite::var2nc::var2nc_4d_dp (
    character(len = *), intent(in) f_name,
    real(dp), dimension(:, :, :, :, :), intent(in) arr,
    character(len = *), dimension(:), intent(in) dnames,
    character(len = *), intent(in) v_name,
    integer(i4), intent(in), optional dim_unlimited,
    character(len = *), intent(in), optional long_name,
    character(len = *), intent(in), optional units,
    real(dp), intent(in), optional missing_value,
    character(256), dimension(:, :, :), intent(in), optional attributes,
    logical, intent(in), optional create,
    integer(i4), intent(inout), optional ncid,
    integer(i4), intent(in), optional nrec )

```

16.122.2.11 var2nc_4d_i4()

```

subroutine mo_ncwrite::var2nc::var2nc_4d_i4 (
    character(len = *), intent(in) f_name,
    integer(i4), dimension(:, :, :, :, :), intent(in) arr,
    character(len = *), dimension(:), intent(in) dnames,
    character(len = *), intent(in) v_name,
    integer(i4), intent(in), optional dim_unlimited,

```

```

character(len = *), intent(in), optional long_name,
character(len = *), intent(in), optional units,
integer(i4), intent(in), optional missing_value,
character(256), dimension(:, :, :), intent(in), optional attributes,
logical, intent(in), optional create,
integer(i4), intent(inout), optional ncid,
integer(i4), intent(in), optional nrec )

```

16.122.2.12 var2nc_4d_sp()

```

subroutine mo_ncwrite::var2nc::var2nc_4d_sp (
    character(len = *), intent(in) f_name,
    real(sp), dimension(:, :, :, :, :), intent(in) arr,
    character(len = *), dimension(:, :), intent(in) dnames,
    character(len = *), intent(in) v_name,
    integer(i4), intent(in), optional dim_unlimited,
    character(len = *), intent(in), optional long_name,
    character(len = *), intent(in), optional units,
    real(sp), intent(in), optional missing_value,
    character(256), dimension(:, :, :), intent(in), optional attributes,
    logical, intent(in), optional create,
    integer(i4), intent(inout), optional ncid,
    integer(i4), intent(in), optional nrec )

```

16.122.2.13 var2nc_5d_dp()

```

subroutine mo_ncwrite::var2nc::var2nc_5d_dp (
    character(len = *), intent(in) f_name,
    real(dp), dimension(:, :, :, :, :, :), intent(in) arr,
    character(len = *), dimension(:, :), intent(in) dnames,
    character(len = *), intent(in) v_name,
    integer(i4), intent(in), optional dim_unlimited,
    character(len = *), intent(in), optional long_name,
    character(len = *), intent(in), optional units,
    real(dp), intent(in), optional missing_value,
    character(256), dimension(:, :, :), intent(in), optional attributes,
    logical, intent(in), optional create,
    integer(i4), intent(inout), optional ncid,
    integer(i4), intent(in), optional nrec )

```

16.122.2.14 var2nc_5d_i4()

```

subroutine mo_ncwrite::var2nc::var2nc_5d_i4 (
    character(len = *), intent(in) f_name,
    integer(i4), dimension(:, :, :, :, :, :), intent(in) arr,
    character(len = *), dimension(:, :), intent(in) dnames,
    character(len = *), intent(in) v_name,
    integer(i4), intent(in), optional dim_unlimited,
    character(len = *), intent(in), optional long_name,
    character(len = *), intent(in), optional units,
    integer(i4), intent(in), optional missing_value,
    character(256), dimension(:, :, :), intent(in), optional attributes,

```

```
logical, intent(in), optional create,
integer(i4), intent(inout), optional ncid,
integer(i4), intent(in), optional nrec )
```

16.122.2.15 var2nc_5d_sp()

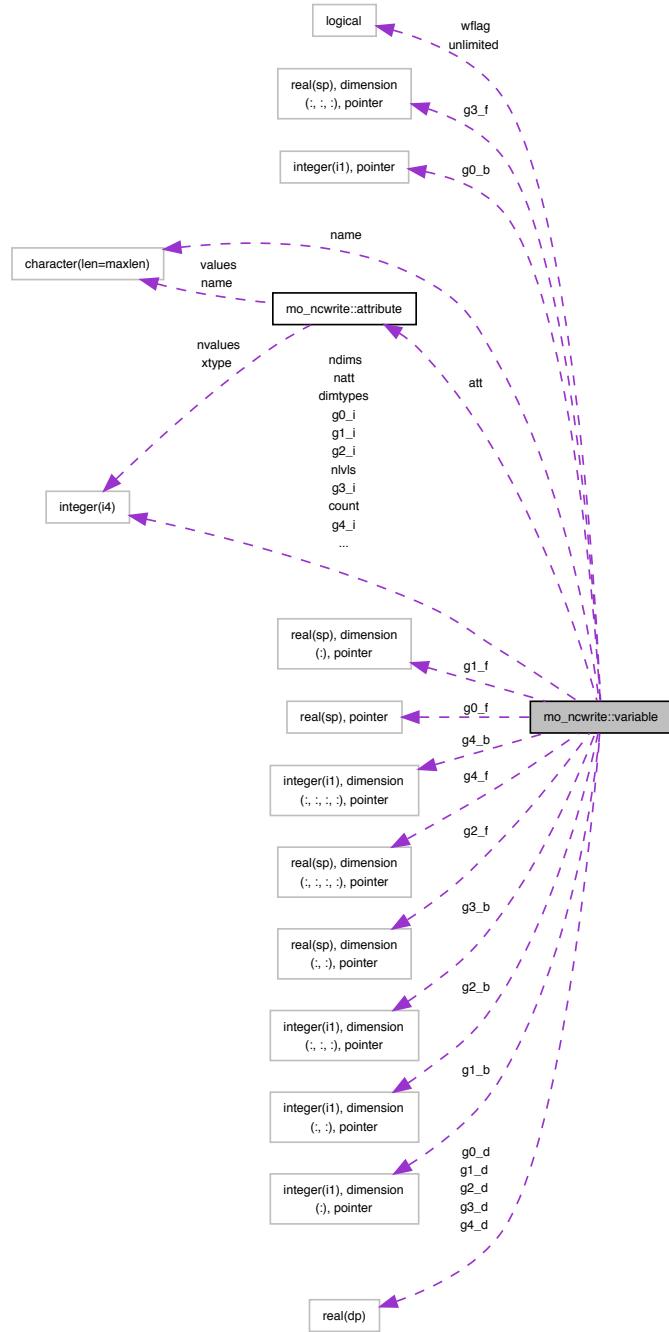
```
subroutine mo_ncwrite::var2nc::var2nc_5d_sp (
    character(len = *), intent(in) f_name,
    real(sp), dimension(:, :, :, :, :), intent(in) arr,
    character(len = *), dimension(:), intent(in) dnames,
    character(len = *), intent(in) v_name,
    integer(i4), intent(in), optional dim_unlimited,
    character(len = *), intent(in), optional long_name,
    character(len = *), intent(in), optional units,
    real(sp), intent(in), optional missing_value,
    character(256), dimension(:, :), intent(in), optional attributes,
    logical, intent(in), optional create,
    integer(i4), intent(inout), optional ncid,
    integer(i4), intent(in), optional nrec )
```

The documentation for this interface was generated from the following file:

- [mo_ncwrite.f90](#)

16.123 mo_ncwrite::variable Type Reference

Collaboration diagram for mo_ncwrite::variable:



Public Attributes

- `character(len=maxlen) name`
- `integer(i4) xtype`
- `integer(i4) nvlis`

- 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**

16.123.1 Member Data Documentation

16.123.1.1 att

```
type(attribute), dimension(nmaxatt) mo_ncwrite::variable::att
```

16.123.1.2 count

```
integer(i4), dimension(nmaxdim) mo_ncwrite::variable::count
```

16.123.1.3 dimids

```
integer(i4), dimension(nmaxdim) mo_ncwrite::variable::dimids
```

16.123.1.4 dimtypes

```
integer(i4), dimension(nmaxdim) mo_ncwrite::variable::dimtypes
```

16.123.1.5 g0_b

```
integer(i1), pointer mo_ncwrite::variable::g0_b
```

16.123.1.6 g0_d

```
real(dp), pointer mo_ncwrite::variable::g0_d
```

16.123.1.7 g0_f

```
real(sp), pointer mo_ncwrite::variable::g0_f
```

16.123.1.8 g0_i

```
integer(i4), pointer mo_ncwrite::variable::g0_i
```

16.123.1.9 g1_b

```
integer(i1), dimension(:), pointer mo_ncwrite::variable::g1_b
```

16.123.1.10 g1_d

```
real(dp), dimension(:), pointer mo_ncwrite::variable::g1_d
```

16.123.1.11 g1_f

```
real(sp), dimension(:), pointer mo_ncwrite::variable::g1_f
```

16.123.1.12 g1_i

```
integer(i4), dimension(:), pointer mo_ncwrite::variable::g1_i
```

16.123.1.13 g2_b

```
integer(i1), dimension(:, :, ), pointer mo_ncwrite::variable::g2_b
```

16.123.1.14 g2_d

```
real(dp), dimension(:, :, ), pointer mo_ncwrite::variable::g2_d
```

16.123.1.15 g2_f

```
real(sp), dimension(:, :, ), pointer mo_ncwrite::variable::g2_f
```

16.123.1.16 g2_i

```
integer(i4), dimension(:, :, ), pointer mo_ncwrite::variable::g2_i
```

16.123.1.17 g3_b

```
integer(i1), dimension(:, :, :, ), pointer mo_ncwrite::variable::g3_b
```

16.123.1.18 g3_d

```
real(dp), dimension(:, :, :, ), pointer mo_ncwrite::variable::g3_d
```

16.123.1.19 g3_f

```
real(sp), dimension(:, :, :, ), pointer mo_ncwrite::variable::g3_f
```

16.123.1.20 g3_i

```
integer(i4), dimension(:, :, :, ), pointer mo_ncwrite::variable::g3_i
```

16.123.1.21 g4_b

```
integer(i1), dimension(:, :, :, :, ), pointer mo_ncwrite::variable::g4_b
```

16.123.1.22 g4_d

```
real(dp), dimension(:, :, :, :, :), pointer mo_ncwrite::variable::g4_d
```

16.123.1.23 g4_f

```
real(sp), dimension(:, :, :, :, :), pointer mo_ncwrite::variable::g4_f
```

16.123.1.24 g4_i

```
integer(i4), dimension(:, :, :, :, :), pointer mo_ncwrite::variable::g4_i
```

16.123.1.25 name

```
character (len = maxlen) mo_ncwrite::variable::name
```

16.123.1.26 natt

```
integer(i4) mo_ncwrite::variable::natt
```

16.123.1.27 ndims

```
integer(i4) mo_ncwrite::variable::ndims
```

16.123.1.28 nlvls

```
integer(i4) mo_ncwrite::variable::nlvls
```

16.123.1.29 nsubs

```
integer(i4) mo_ncwrite::variable::nsubs
```

16.123.1.30 start

```
integer(i4), dimension(nmaxdim) mo_ncwrite::variable::start
```

16.123.1.31 unlimited

```
logical mo_ncwrite::variable::unlimited
```

16.123.1.32 varid

```
integer(i4) mo_ncwrite::variable::varid
```

16.123.1.33 wflag

```
logical mo_ncwrite::variable::wflag
```

16.123.1.34 xtype

```
integer(i4) mo_ncwrite::variable::xtype
```

The documentation for this type was generated from the following file:

- [mo_ncwrite.f90](#)

16.124 mo_moment::variance Interface Reference

Public Member Functions

- `real(sp) function variance_sp (dat, mask)`
- `real(dp) function variance_dp (dat, mask)`

16.124.1 Member Function/Subroutine Documentation

16.124.1.1 variance_dp()

```
real(dp) function mo_moment::variance::variance_dp (
    real(dp), dimension(:), intent(in) dat,
    logical, dimension(:), intent(in), optional mask )
```

16.124.1.2 variance_sp()

```
real(sp) function mo_moment::variance::variance_sp (
    real(sp), dimension(:), intent(in) dat,
    logical, dimension(:), intent(in), optional mask )
```

The documentation for this interface was generated from the following file:

- [mo_moment.f90](#)

16.125 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)

16.125.1 Member Function/Subroutine Documentation

16.125.1.1 `wnse_dp_1d()`

```
real(dp) function mo_errormeasures::wnse:::wnse_dp_1d (
    real(dp), dimension(:), intent(in) x,
    real(dp), dimension(:), intent(in) y,
    logical, dimension(:), intent(in), optional mask )
```

16.125.1.2 `wnse_dp_2d()`

```
real(dp) function mo_errormeasures::wnse:::wnse_dp_2d (
    real(dp), dimension(:, :, :), intent(in) x,
    real(dp), dimension(:, :, :), intent(in) y,
    logical, dimension(:, :, :), intent(in), optional mask )
```

16.125.1.3 `wnse_dp_3d()`

```
real(dp) function mo_errormeasures::wnse:::wnse_dp_3d (
    real(dp), dimension(:, :, :, :), intent(in) x,
    real(dp), dimension(:, :, :, :), intent(in) y,
    logical, dimension(:, :, :, :), intent(in), optional mask )
```

16.125.1.4 `wnse_sp_1d()`

```
real(sp) function mo_errormeasures::wnse:::wnse_sp_1d (
    real(sp), dimension(:), intent(in) x,
    real(sp), dimension(:), intent(in) y,
    logical, dimension(:), intent(in), optional mask )
```

16.125.1.5 `wnse_sp_2d()`

```
real(sp) function mo_errormeasures::wnse:::wnse_sp_2d (
    real(sp), dimension(:, :, :), intent(in) x,
```

```
real(sp), dimension(:,:), intent(in) y,
logical, dimension(:,:), intent(in), optional mask )
```

16.125.1.6 `wnse_sp_3d()`

```
real(sp) function mo_errormeasures::wnse::wnse_sp_3d (
    real(sp), dimension(:,:,:), intent(in) x,
    real(sp), dimension(:,:,:), intent(in) y,
    logical, dimension(:,:,:), intent(in), optional mask )
```

The documentation for this interface was generated from the following file:

- [mo_errormeasures.f90](#)

16.126 `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)

16.126.1 Member Function/Subroutine Documentation

16.126.1.1 `xor4096d_0d()`

```
subroutine mo_xor4096::xor4096::xor4096d_0d (
    integer(i8), intent(in) seed,
    real(dp), intent(out) DoubleRealRN,
    integer(i8), dimension(n\_save\_state), intent(inout), optional save_state )
```

16.126.1.2 `xor4096d_1d()`

```
subroutine mo_xor4096::xor4096::xor4096d_1d (
    integer(i8), dimension(:), intent(in) seed,
    real(dp), dimension(size(seed, 1)), intent(out) DoubleRealRN,
    integer(i8), dimension(size(seed, 1), n\_save\_state), intent(inout), optional
    save_state )
```

16.126.1.3 xor4096f_0d()

```
subroutine mo_xor4096::xor4096::xor4096f_0d (
    integer(i4), intent(in) seed,
    real(sp), intent(out) SingleRealRN,
    integer(i4), dimension(n\_save\_state), intent(inout), optional save_state )
```

16.126.1.4 xor4096f_1d()

```
subroutine mo_xor4096::xor4096::xor4096f_1d (
    integer(i4), dimension(:), intent(in) seed,
    real(sp), dimension(size(seed)), intent(out) SingleRealRN,
    integer(i4), dimension(size(seed, 1), n\_save\_state), intent(inout), optional
    save_state )
```

16.126.1.5 xor4096l_0d()

```
subroutine mo_xor4096::xor4096::xor4096l_0d (
    integer(i8), intent(in) seed,
    integer(i8), intent(out) DoubleIntegerRN,
    integer(i8), dimension(n\_save\_state), intent(inout), optional save_state )
```

16.126.1.6 xor4096l_1d()

```
subroutine mo_xor4096::xor4096::xor4096l_1d (
    integer(i8), dimension(:), intent(in) seed,
    integer(i8), dimension(size(seed, 1)), intent(out) DoubleIntegerRN,
    integer(i8), dimension(size(seed, 1), n\_save\_state), intent(inout), optional
    save_state )
```

16.126.1.7 xor4096s_0d()

```
subroutine mo_xor4096::xor4096::xor4096s_0d (
    integer(i4), intent(in) seed,
    integer(i4), intent(out) SingleIntegerRN,
    integer(i4), dimension(n\_save\_state), intent(inout), optional save_state )
```

16.126.1.8 xor4096s_1d()

```
subroutine mo_xor4096::xor4096::xor4096s_1d (
    integer(i4), dimension(:), intent(in) seed,
    integer(i4), dimension(size(seed, 1)), intent(out) SingleIntegerRN,
    integer(i4), dimension(size(seed, 1), n\_save\_state), intent(inout), optional
    save_state )
```

The documentation for this interface was generated from the following file:

- [mo_xor4096.f90](#)

16.127 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)

16.127.1 Member Function/Subroutine Documentation

16.127.1.1 xor4096gd_0d()

```
subroutine mo_xor4096::xor4096g::xor4096gd_0d (
    integer(i8), intent(in) seed,
    real(dp), intent(out) DoubleRealRN,
    integer(i8), dimension(n\_save\_state), intent(inout), optional save_state )
```

16.127.1.2 xor4096gd_1d()

```
subroutine mo_xor4096::xor4096g::xor4096gd_1d (
    integer(i8), dimension(:), intent(in) seed,
    real(dp), dimension(size(seed)), intent(out) DoubleRealRN,
    integer(i8), dimension(size(seed), n\_save\_state), intent(inout), optional save_←
state )
```

16.127.1.3 xor4096gf_0d()

```
subroutine mo_xor4096::xor4096g::xor4096gf_0d (
    integer(i4), intent(in) seed,
    real(sp), intent(out) SingleRealRN,
    integer(i4), dimension(n\_save\_state), intent(inout), optional save_state )
```

16.127.1.4 xor4096gf_1d()

```
subroutine mo_xor4096::xor4096g::xor4096gf_1d (
    integer(i4), dimension(:), intent(in) seed,
    real(sp), dimension(size(seed)), intent(out) SingleRealRN,
    integer(i4), dimension(size(seed), n\_save\_state), intent(inout), optional save_←
state )
```

The documentation for this interface was generated from the following file:

- [mo_xor4096.f90](#)

Chapter 17

File Documentation

17.1 1-main.dox File Reference

17.2 2-get_started.dox File Reference

17.3 3-data_preparation.dox File Reference

17.4 4-visualise_out.dox File Reference

17.5 5-calibration.dox File Reference

17.6 6-style_guide.dox File Reference

17.7 7-test_basin.dox File Reference

17.8 8-protocols_for_setup_new_mHM_basin.dox File Reference

17.9 DEPENDENCIES.md File Reference

17.10 mhm_driver.f90 File Reference

Functions/Subroutines

- program [mhm_driver](#)

Distributed precipitation-runoff model mHM.

17.10.1 Function/Subroutine Documentation

17.10.1.1 mhm_driver()

```
program mhm_driver ( )
```

Distributed precipitation-runoff model mHM.

This is the main driver of mHM, which calls one instance of mHM for a multiple basins and a given period.

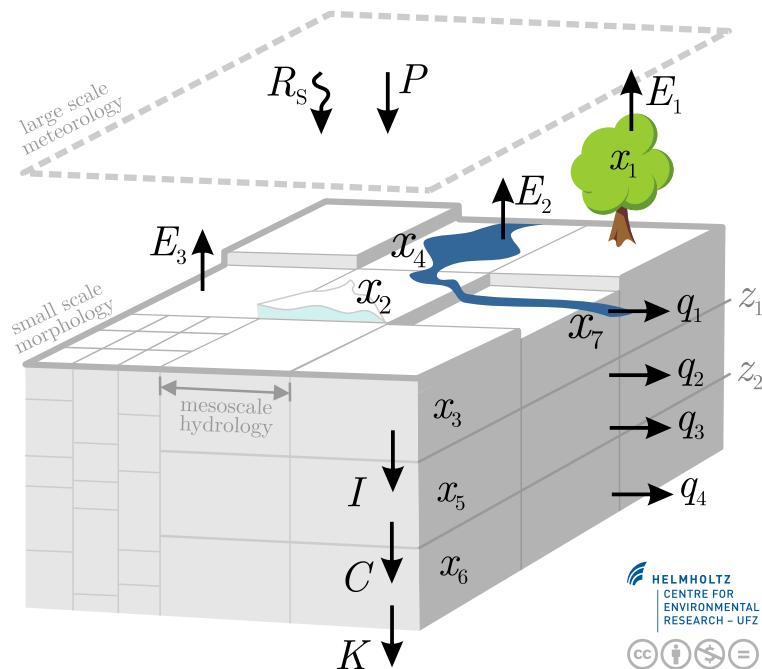


Figure 17.1: Typical mHM cell

Luis Samaniego & Rohini Kumar (UFZ)

Date

Jun 2018

Version

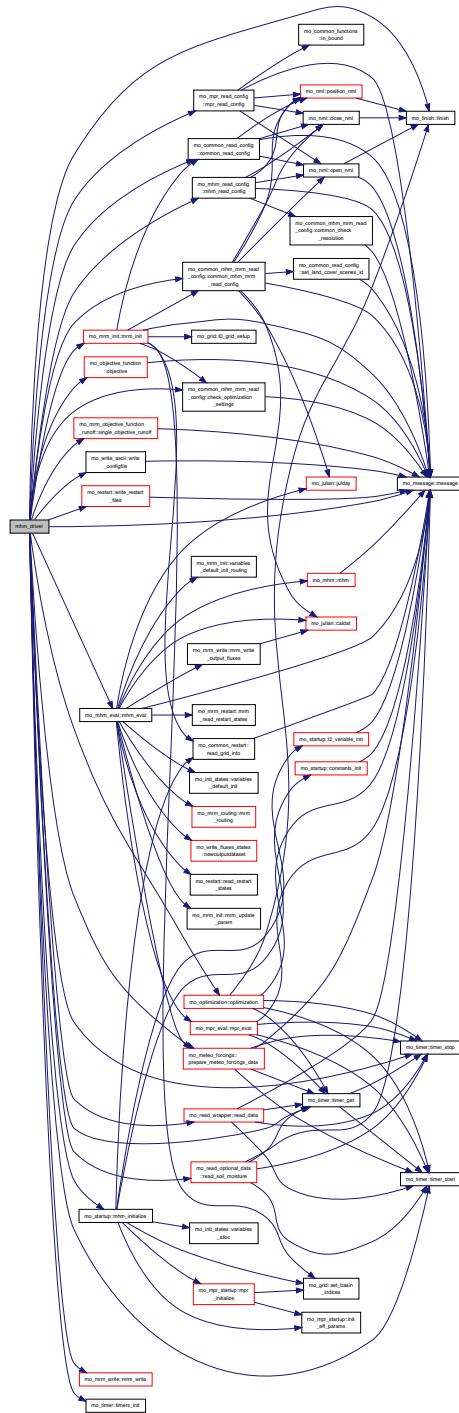
5.9

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 DEVELOPERS MAKES ANY WARRANTY, EXPRESS OR IMPLIED, OR ASSUMES ANY LIABILITY FOR THE 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.

References mo_common_mhm_mrm_read_config::check_optimization_settings(), mo_common_mhm_mrm::read_config::common_mhm_mrm_read_config(), mo_common_read_config::common_read_config(), mo_global::variables::dirprecipitation, mo_kind::dp, mo_file::file_main, mo_finish::finish(), mo_kind::i4, mo_message::message(), mo_message::message_text, mo_mhm_eval::mhm_eval(), mo_startup::mhm_initialize(), mo_mhm::read_config::mhm_read_config(), mo_mpr_read_config::mpr_read_config(), mo_mrm_init::mrm_init(), mo_mrm::write::mrm_write(), mo_common_mhm_mrm_variables::ntstepday, mo_objective_function::objective(), mo_optimization::optimization(), mo_meteo_forcings::prepare_meteo_forcings_data(), mo_read_wrapper::read_data(), mo_read_optional_data::read_soil_moisture(), mo_string_utils::separator, mo_mrm_objective_function_runoff::single_objective_runoff(), mo_timer::timer_get(), mo_timer::timer_start(), mo_timer::timer_stop(), mo_timer::timers_init(), mo_file::version, mo_file::version_date, mo_write_ascii::write_configfile(), mo_common_variables::write_restart, and mo_restart::write_restart_files().

Here is the call graph for this function:



17.11 mhm_papers.md File Reference

17.12 mo_anneal.f90 File Reference

Data Types

- interface `mo_anneal::anneal`
`anneal`
- interface `mo_anneal::gettemperature`
`GetTemperature.`
- interface `mo_anneal::generate_neighborhood_weight`

Modules

- module `mo_anneal`

Functions/Subroutines

- real(dp) function, dimension(size(para, 1)) `mo_anneal::anneal_dp` (eval, cost, para, prange, prange_func, temp, Dt, nITERmax, Len, nST, eps, acc, seeds, printflag, maskpara, weight, changeParaMode, reflection←Flag, pertubFlexFlag, maxit, undef_funcval, tmp_file, funcbest, history)
- real(dp) function `mo_anneal::gettemperature_dp` (paraset, cost, eval, acc_goal, prange, prange_func, samplesize, maskpara, seeds, printflag, weight, maxit, undef_funcval)
- real(dp) function `mo_anneal::pargen_anneal_dp` (old, dMax, oMin, oMax, RN)
- real(dp) function `mo_anneal::pargen_dds_dp` (old, perturb, oMin, oMax, RN)
- real(dp) function `mo_anneal::dchange_dp` (delta, iDigit, isZero)
- subroutine `mo_anneal::generate_neighborhood_weight_dp` (truepara, cum_weight, save_state_xor, iTotal←Counter, nITERmax, neighborhood)

17.13 mo_append.f90 File Reference

Data Types

- interface `mo_append::append`
`Append (rows) scalars, vectors, and matrixes onto existing array.`
- interface `mo_append::paste`
`Paste (columns) scalars, vectors, and matrixes onto existing array.`

Modules

- module `mo_append`
`Append values on existing arrays.`

Functions/Subroutines

- subroutine `mo_append::append_i4_v_s` (vec1, sca2)
- subroutine `mo_append::append_i4_v_v` (vec1, vec2)
- subroutine `mo_append::append_i4_m_m` (mat1, mat2, fill_value)
- subroutine `mo_append::append_i4_3d` (mat1, mat2, fill_value)
- subroutine `mo_append::append_i8_v_s` (vec1, sca2)
- subroutine `mo_append::append_i8_v_v` (vec1, vec2)
- subroutine `mo_append::append_i8_m_m` (mat1, mat2, fill_value)
- subroutine `mo_append::append_i8_3d` (mat1, mat2, fill_value)
- subroutine `mo_append::append_sp_v_s` (vec1, sca2)
- subroutine `mo_append::append_sp_v_v` (vec1, vec2)

- subroutine `mo_append::append_sp_m_m` (mat1, mat2, fill_value)
- subroutine `mo_append::append_sp_3d` (mat1, mat2, fill_value)
- subroutine `mo_append::append_dp_v_s` (vec1, sca2)
- subroutine `mo_append::append_dp_v_v` (vec1, vec2)
- subroutine `mo_append::append_dp_m_m` (mat1, mat2, fill_value)
- subroutine `mo_append::append_dp_3d` (mat1, mat2, fill_value)
- subroutine `mo_append::append_char_v_s` (vec1, sca2)
- subroutine `mo_append::append_char_v_v` (vec1, vec2)
- subroutine `mo_append::append_char_m_m` (mat1, mat2, fill_value)
- subroutine `mo_append::append_char_3d` (mat1, mat2, fill_value)
- subroutine `mo_append::append_lgt_v_s` (vec1, sca2)
- subroutine `mo_append::append_lgt_v_v` (vec1, vec2)
- subroutine `mo_append::append_lgt_m_m` (mat1, mat2, fill_value)
- subroutine `mo_append::append_lgt_3d` (mat1, mat2, fill_value)
- subroutine `mo_append::paste_i4_m_s` (mat1, sca2)
- subroutine `mo_append::paste_i4_m_v` (mat1, vec2, fill_value)
- subroutine `mo_append::paste_i4_m_m` (mat1, mat2, fill_value)
- subroutine `mo_append::paste_i8_m_s` (mat1, sca2)
- subroutine `mo_append::paste_i8_m_v` (mat1, vec2, fill_value)
- subroutine `mo_append::paste_i8_m_m` (mat1, mat2, fill_value)
- subroutine `mo_append::paste_sp_m_s` (mat1, sca2)
- subroutine `mo_append::paste_sp_m_v` (mat1, vec2, fill_value)
- subroutine `mo_append::paste_sp_m_m` (mat1, mat2, fill_value)
- subroutine `mo_append::paste_dp_m_s` (mat1, sca2)
- subroutine `mo_append::paste_dp_m_v` (mat1, vec2, fill_value)
- subroutine `mo_append::paste_dp_m_m` (mat1, mat2, fill_value)
- subroutine `mo_append::paste_char_m_s` (mat1, sca2)
- subroutine `mo_append::paste_char_m_v` (mat1, vec2, fill_value)
- subroutine `mo_append::paste_char_m_m` (mat1, mat2, fill_value)
- subroutine `mo_append::paste_lgt_m_s` (mat1, sca2)
- subroutine `mo_append::paste_lgt_m_v` (mat1, vec2)
- subroutine `mo_append::paste_lgt_m_m` (mat1, mat2)

17.14 mo_canopy_interc.f90 File Reference

Modules

- module `mo_canopy_interc`
Canopy interception.

Functions/Subroutines

- elemental pure subroutine, public `mo_canopy_interc::canopy_interc` (pet, interc_max, precip, interc, through-fall, evap_canopy)
Canopy interception.

17.15 mo_common_constants.f90 File Reference

Modules

- module `mo_common_constants`
Provides constants commonly used by mHM, mRM and MPR.

Variables

- real(dp), parameter, public `mo_common_constants::eps_dp` = `epsilon(1.0_dp)`
epsilon(1.0) in double precision
- real(sp), parameter, public `mo_common_constants::eps_sp` = `epsilon(1.0_sp)`
epsilon(1.0) in single precision
- integer(i4), parameter, public `mo_common_constants::nodata_i4` = -9999_i4
- real(dp), parameter, public `mo_common_constants::nodata_dp` = -9999_dp
- real(dp), parameter, public `mo_common_constants::p1_initstatefluxes` = 0.00_dp
- integer(i4), parameter, public `mo_common_constants::ncolpars` = 5_i4
- integer(i4), parameter, public `mo_common_constants::maxnobasins` = 50_i4
- integer(i4), parameter, public `mo_common_constants::maxnlcovers` = 50_i4
- real(dp), parameter, public `mo_common_constants::dayhours` = 24.0_dp
- real(dp), parameter, public `mo_common_constants::yearmonths` = 12.0_dp
- integer(i4), parameter, public `mo_common_constants::yearmonths_i4` = 12
- real(dp), parameter, public `mo_common_constants::yeardays` = 365.0_dp
- real(dp), parameter, public `mo_common_constants::daysecs` = 86400.0_dp
- real(dp), parameter, public `mo_common_constants::hoursecs` = 3600.0_dp

17.16 mo_common_file.f90 File Reference

Modules

- module `mo_common_file`
Provides file names and units for mRM.

Variables

- character(len= *), parameter `mo_common_file::file_dem` = 'dem.asc'
DEM input data file.
- integer, parameter `mo_common_file::udem` = 53
Unit for DEM input data file.
- integer, parameter `mo_common_file::ulcoverclass` = 61
Unit for LCover input data file.
- character(len= *), parameter `mo_common_file::file_config` = 'ConfigFile.log'
file defining mHM's outputs
- integer, parameter `mo_common_file::uconfig` = 68
Unit for file defining mHM's outputs.

17.17 mo_common_functions.f90 File Reference

Modules

- module `mo_common_functions`
Provides small utility functions used by multiple parts of the code (mHM, mRM, MPR)

Functions/Subroutines

- logical function, public `mo_common_functions::in_bound` (params)
TODO: add description.

17.18 mo_common_mHM_mRM_file.f90 File Reference

Modules

- module [mo_common_mhm_mrm_file](#)

Provides file names and units for mHM.

Variables

- character(len=*), parameter [mo_common_mhm_mrm_file::file_opti](#) = 'FinalParam.out'
file defining optimization outputs (objective and parameter set)
- integer, parameter [mo_common_mhm_mrm_file::uopti](#) = 72
Unit for file optimization outputs (objective and parameter set)
- character(len=*), parameter [mo_common_mhm_mrm_file::file_opti_nml](#) = 'FinalParam.nml'
file defining optimization outputs in a namelist format (parameter set)
- integer, parameter [mo_common_mhm_mrm_file::uopti_nml](#) = 73
Unit for file optimization outputs in a namelist format (parameter set)

17.19 mo_common_mHM_mRM_read_config.f90 File Reference

Modules

- module [mo_common_mhm_mrm_read_config](#)

Reading of main model configurations.

Functions/Subroutines

- subroutine, public [mo_common_mhm_mrm_read_config::common_mhm_mrm_read_config](#) (file_namelist, unamelist)
Read main configurations for common parts.
- subroutine, public [mo_common_mhm_mrm_read_config::check_optimization_settings](#)
TODO: add description.
- subroutine, public [mo_common_mhm_mrm_read_config::common_check_resolution](#) (do_message, allow←_subgrid_routing)
TODO: add description.

17.20 mo_common_mHM_mRM_variables.f90 File Reference

Modules

- module [mo_common_mhm_mrm_variables](#)

Provides structures needed by mHM, mRM and/or mpr.

Variables

- integer(i4) [mo_common_mhm_mrm_variables::mrm_coupling_mode](#)
- integer(i4), public [mo_common_mhm_mrm_variables::timestep](#)
- real(dp), dimension(:), allocatable, public [mo_common_mhm_mrm_variables::resolutionrouting](#)
- logical, public [mo_common_mhm_mrm_variables::read_restart](#)

- type(period), dimension(:), allocatable, public `mo_common_mhm_mrm_variables::warmper`
- type(period), dimension(:), allocatable, public `mo_common_mhm_mrm_variables::evalper`
- type(period), dimension(:), allocatable, public `mo_common_mhm_mrm_variables::simper`
- type(period), public `mo_common_mhm_mrm_variables::readper`
- integer(i4), dimension(:), allocatable, public `mo_common_mhm_mrm_variables::warmingdays`
- integer(i4), dimension(:, :), allocatable, public `mo_common_mhm_mrm_variables::lcyearid`
- integer(i4), public `mo_common_mhm_mrm_variables::ntstepday`
- character(256), dimension(:), allocatable, public `mo_common_mhm_mrm_variables::dirrestartin`
- integer(i4), public `mo_common_mhm_mrm_variables::opti_method`
- integer(i4), public `mo_common_mhm_mrm_variables::opti_function`
- logical, public `mo_common_mhm_mrm_variables::optimize`
- logical, public `mo_common_mhm_mrm_variables::optimize_restart`
- integer(i8), public `mo_common_mhm_mrm_variables::seed`
- integer(i4), public `mo_common_mhm_mrm_variables::niterations`
- real(dp), public `mo_common_mhm_mrm_variables::dds_r`
- real(dp), public `mo_common_mhm_mrm_variables::sa_temp`
- integer(i4), public `mo_common_mhm_mrm_variables::sce_ngs`
- integer(i4), public `mo_common_mhm_mrm_variables::sce_npg`
- integer(i4), public `mo_common_mhm_mrm_variables::sce_nps`
- logical, public `mo_common_mhm_mrm_variables::mcmc_opti`
- integer(i4), parameter, public `mo_common_mhm_mrm_variables::nerror_model = 2`
- real(dp), dimension(nerror_model), public `mo_common_mhm_mrm_variables::mcmc_error_params`

17.21 mo_common_read_config.f90 File Reference

Modules

- module `mo_common_read_config`
Reading of main model configurations.

Functions/Subroutines

- subroutine, public `mo_common_read_config::common_read_config` (file_namelist, unamelist)
Read main configurations commonly used by mHM, mRM and MPR.
- subroutine, public `mo_common_read_config::set_land_cover_scenes_id` (sim_Per, LCyear_Id, LCfilename)
Read main configurations commonly used by mHM, mRM and MPR.

17.22 mo_common_read_data.f90 File Reference

Modules

- module `mo_common_read_data`
TODO: add description.

Functions/Subroutines

- subroutine, public `mo_common_read_data::read_dem`
TODO: add description.
- subroutine, public `mo_common_read_data::read_lcover`
TODO: add description.

17.23 mo_common_restart.f90 File Reference

Modules

- module `mo_common_restart`

TODO: add description.

Functions/Subroutines

- subroutine, public `mo_common_restart::write_grid_info` (`grid_in`, `level_name`, `nc`)
write restart files for each basin
- subroutine, public `mo_common_restart::read_grid_info` (`iBasin`, `InPath`, `level_name`, `fname_part`, `new_grid`)
reads configuration apart from Level 11 configuration from a restart directory

17.24 mo_common_variables.f90 File Reference

Data Types

- type `mo_common_variables::period`
- type `mo_common_variables::grid`
- type `mo_common_variables::gridremapper`

Modules

- module `mo_common_variables`

Provides structures needed by mHM, mRM and/or mpr.

Variables

- character(1024), public `mo_common_variables::project_details`
- character(1024), public `mo_common_variables::setup_description`
- character(1024), public `mo_common_variables::simulation_type`
- character(256), public `mo_common_variables::conventions`
- character(1024), public `mo_common_variables::contact`
- character(1024), public `mo_common_variables::mhm_details`
- character(1024), public `mo_common_variables::history`
- integer(i4), public `mo_common_variables::iflag_coordinate_sys`
- real(dp), dimension(:), allocatable, public `mo_common_variables::resolutionhydrology`
- integer(i4), dimension(:), allocatable, public `mo_common_variables::l0_basin`
- logical, public `mo_common_variables::write_restart`
- character(256), dimension(:), allocatable, public `mo_common_variables::dirrestartout`
- character(256), public `mo_common_variables::dirconfigout`
- character(256), public `mo_common_variables::dircommonfiles`
- character(256), dimension(:), allocatable, public `mo_common_variables::dirmorpho`
- character(256), dimension(:), allocatable, public `mo_common_variables::dircover`
- character(256), dimension(:), allocatable, public `mo_common_variables::dirout`
- character(256), dimension(:), allocatable, public `mo_common_variables::filelatlon`
- type(grid), dimension(:), allocatable, target, public `mo_common_variables::level0`
- type(grid), dimension(:), allocatable, target, public `mo_common_variables::level1`
- type(gridremapper), dimension(:), allocatable, public `mo_common_variables::l0_l1_remap`
- real(dp), dimension(:), allocatable, public `mo_common_variables::l0_elev`

- integer(i4), dimension(:, :,), allocatable, public `mo_common_variables::l0_lcover`
 - integer(i4), public `mo_common_variables::nbasins`
 - integer(i4), public `mo_common_variables::nunique0basins`
 - integer(i4), public `mo_common_variables::nlcoverscene`
 - character(256), dimension(:,), allocatable, public `mo_common_variables::lcfilename`
 - integer(i4), dimension(:,), allocatable, public `mo_common_variables::lc_year_start`
 - integer(i4), dimension(:,), allocatable, public `mo_common_variables::lc_year_end`
 - integer(i4), parameter, public `mo_common_variables::nprocesses` = 10
 - integer(i4), dimension(nprocesses, 3), public `mo_common_variables::processmatrix`
 - real(dp), dimension(:, :,), allocatable, target, public `mo_common_variables::global_parameters`
 - character(256), dimension(:,), allocatable, public `mo_common_variables::global_parameters_name`
 - logical `mo_common_variables::alma_convention`

17.25 mo_constants.f90 File Reference

Modules

- module `mo_constants`
Provides computational, mathematical, physical, and file constants.

Variables

- `real(dp)`, parameter `mo_constants::pi_dp` = 3.141592653589793238462643383279502884197_dp
Pi in double precision.
 - `real(sp)`, parameter `mo_constants::pi_sp` = 3.141592653589793238462643383279502884197_sp
Pi in single precision.
 - `real(dp)`, parameter `mo_constants::pi2_dp` = 1.57079632679489661923132169163975144209858_dp
Pi/2 in double precision.
 - `real(sp)`, parameter `mo_constants::pi2_sp` = 1.57079632679489661923132169163975144209858_sp
Pi/2 in single precision.
 - `real(dp)`, parameter `mo_constants::twopi_dp` = 6.283185307179586476925286766559005768394_dp
*2*Pi in double precision*
 - `real(sp)`, parameter `mo_constants::twopi_sp` = 6.283185307179586476925286766559005768394_sp
*2*Pi in single precision*
 - `real(dp)`, parameter `mo_constants::sqrt2_dp` = 1.41421356237309504880168872420969807856967_dp
Square root of 2 in double precision.
 - `real(sp)`, parameter `mo_constants::sqrt2_sp` = 1.41421356237309504880168872420969807856967_sp
Square root of 2 in single precision.
 - `real(dp)`, parameter `mo_constants::twothird_dp` = 0.667_dp
2/3 in double precision
 - `real(sp)`, parameter `mo_constants::twothird_sp` = 0.667_sp
2/3 in single precision
 - `real(dp)`, parameter `mo_constants::deg2rad_dp` = PI_dp / 180._dp
degree to radian conversion (pi/180) in double precision
 - `real(sp)`, parameter `mo_constants::deg2rad_sp` = PI_sp / 180._sp
degree to radian conversion (pi/180) in double precision
 - `real(dp)`, parameter `mo_constants::rad2deg_dp` = 180._dp / PI_dp
radian to conversion (180/pi) in double precision
 - `real(sp)`, parameter `mo_constants::rad2deg_sp` = 180._sp / PI_sp
radian to degree conversion (180/pi) in single precision

- real(sp), parameter, public `mo_constants::secday_sp` = 86400.0_sp
Time conversion Seconds per day [s] in single precision.
- real(dp), parameter, public `mo_constants::secday_dp` = 86400.0_dp
- real(dp), parameter, public `mo_constants::dayhours` = 24.0_dp
- real(dp), parameter, public `mo_constants::yeartmonths` = 12.0_dp
- real(dp), parameter, public `mo_constants::yeardays` = 365.0_dp
- real(dp), parameter, public `mo_constants::daysecs` = 86400.0_dp
- real(dp), parameter `mo_constants::psychro_dp` = 0.0646_dp
Psychrometric constant [kPa K^-1] in double precision.
- real(sp), parameter `mo_constants::psychro_sp` = 0.0646_sp
Psychrometric constant [kPa K^-1] in single precision.
- real(dp), parameter `mo_constants::gravity_dp` = 9.81_dp
Gravity acceleration [m^2 s^-1] in double precision.
- real(sp), parameter `mo_constants::gravity_sp` = 9.81_sp
Gravity acceleration [m^2 s^-1] in single precision.
- real(dp), parameter `mo_constants::solarconst_dp` = 1367._dp
Solar constant in [J m^-2 s^-1] in double precision.
- real(sp), parameter `mo_constants::solarconst_sp` = 1367._sp
Solar constant in [J m^-2 s^-1] in single precision.
- real(dp), parameter `mo_constants::specheatet_dp` = 2.45e06_dp
Specific heat for vaporization of water in [J m^-2 mm-1] in double precision.
- real(sp), parameter `mo_constants::specheatet_sp` = 2.45e06_sp
Specific heat for vaporization of water in [J m^-2 mm-1] in single precision.
- real(dp), parameter `mo_constants::t0_dp` = 273.15_dp
Standard temperature [K] in double precision.
- real(sp), parameter `mo_constants::t0_sp` = 273.15_sp
Standard temperature [K] in single precision.
- real(dp), parameter `mo_constants::sigma_dp` = 5.67e-08_dp
Stefan-Boltzmann constant [W m^-2 K^-4] in double precision.
- real(sp), parameter `mo_constants::sigma_sp` = 5.67e-08_sp
Stefan-Boltzmann constant [W m^-2 K^-4] in single precision.
- real(sp), parameter `mo_constants::radiusearth_sp` = 6371228._sp
- real(dp), parameter `mo_constants::radiusearth_dp` = 6371228._dp
- real(dp), parameter `mo_constants::p0_dp` = 101325._dp
standard atmosphere Standard pressure [Pa] in double precision
- real(sp), parameter `mo_constants::p0_sp` = 101325._sp
Standard pressure [Pa] in single precision.
- real(dp), parameter `mo_constants::rho0_dp` = 1.225_dp
standard density [kg m^-3] in double precision
- real(sp), parameter `mo_constants::rho0_sp` = 1.225_sp
standard density [kg m^-3] in single precision
- real(dp), parameter `mo_constants::cp0_dp` = 1005.0_dp
specific heat capacity of air [J kg^-1 K^-1] in double precision
- real(sp), parameter `mo_constants::cp0_sp` = 1005.0_sp
specific heat capacity of air [J kg^-1 K^-1] in single precision
- real(dp), parameter `mo_constants::pi_d` = 3.141592653589793238462643383279502884197_dp
Pi in double precision.
- real(sp), parameter `mo_constants::pi` = 3.141592653589793238462643383279502884197_sp
Pi in single precision.
- real(dp), parameter `mo_constants::pio2_d` = 1.57079632679489661923132169163975144209858_dp
Pi/2 in double precision.

- real(sp), parameter `mo_constants::pio2` = 1.57079632679489661923132169163975144209858_sp
Pi/2 in single precision.
- real(dp), parameter `mo_constants::twopi_d` = 6.283185307179586476925286766559005768394_dp
*2*Pi in double precision*
- real(sp), parameter `mo_constants::twopi` = 6.283185307179586476925286766559005768394_sp
*2*Pi in single precision*
- real(dp), parameter `mo_constants::sqrt2_d` = 1.41421356237309504880168872420969807856967_dp
Square root of 2 in double precision.
- real(sp), parameter `mo_constants::sqrt2` = 1.41421356237309504880168872420969807856967_sp
Square root of 2 in single precision.
- real(dp), parameter `mo_constants::euler_d` = 0.5772156649015328606065120900824024310422_dp
Euler's constant in double precision.
- real(sp), parameter `mo_constants::euler` = 0.5772156649015328606065120900824024310422_sp
Euler's constant in single precision.
- integer, parameter `mo_constants::nin` = `input_unit`
Standard input file unit.
- integer, parameter `mo_constants::nout` = `output_unit`
Standard output file unit.
- integer, parameter `mo_constants::nerr` = `error_unit`
Standard error file unit.
- integer, parameter `mo_constants::nnml` = 100
Standard file unit for namelist.

17.26 mo_corr.f90 File Reference

Data Types

- interface `mo_corr::autocoeffk`
- interface `mo_corr::autocorr`
- interface `mo_corr::corr`
- interface `mo_corr::crosscoeffk`
- interface `mo_corr::crosscorr`
- interface `mo_corr::arth`
- interface `mo_corr::four1`
- interface `mo_corr::fourrow`
- interface `mo_corr::realft`
- interface `mo_corr::swap`

Modules

- module `mo_corr`

Functions/Subroutines

- real(sp) function, dimension(n) `mo_corr::arth_sp` (first, increment, n)
- real(dp) function, dimension(n) `mo_corr::arth_dp` (first, increment, n)
- integer(i4) function, dimension(n) `mo_corr::arth_i4` (first, increment, n)
- real(dp) function `mo_corr::autocoeffk_dp` (x, k, mask)
- real(sp) function `mo_corr::autocoeffk_sp` (x, k, mask)
- real(dp) function, dimension(size(k)) `mo_corr::autocoeffk_1d_dp` (x, k, mask)
- real(sp) function, dimension(size(k)) `mo_corr::autocoeffk_1d_sp` (x, k, mask)

- real(dp) function `mo_corr::autocorr_dp` (x, k, mask)
- real(sp) function `mo_corr::autocorr_sp` (x, k, mask)
- real(dp) function, dimension(size(k)) `mo_corr::autocorr_1d_dp` (x, k, mask)
- real(sp) function, dimension(size(k)) `mo_corr::autocorr_1d_sp` (x, k, mask)
- real(dp) function, dimension(size(data1)) `mo_corr::corr_dp` (data1, data2, nadjust, nhigh, nwin)
- real(sp) function, dimension(size(data1)) `mo_corr::corr_sp` (data1, data2, nadjust, nhigh, nwin)
- real(dp) function `mo_corr::crosscoeffk_dp` (x, y, k, mask)
- real(sp) function `mo_corr::crosscoeffk_sp` (x, y, k, mask)
- real(dp) function `mo_corr::crosscorr_dp` (x, y, k, mask)
- real(sp) function `mo_corr::crosscorr_sp` (x, y, k, mask)
- subroutine `mo_corr::four1_sp` (data, isign)
- subroutine `mo_corr::four1_dp` (data, isign)
- subroutine `mo_corr::fourrow_sp` (data, isign)
- subroutine `mo_corr::fourrow_dp` (data, isign)
- subroutine `mo_corr::realft_dp` (data, isign, zdata)
- subroutine `mo_corr::realft_sp` (data, isign, zdata)
- subroutine `mo_corr::swap_1d_spc` (a, b)
- subroutine `mo_corr::swap_1d_dpc` (a, b)
- complex(dpc) function, dimension(nn) `mo_corr::zroots_unity_dp` (n, nn)
- complex(spc) function, dimension(nn) `mo_corr::zroots_unity_sp` (n, nn)

Variables

- integer(i4), parameter `mo_corr::npar_arth` = 16
- integer(i4), parameter `mo_corr::npar2_arth` = 8

17.27 mo_dds.f90 File Reference

Modules

- module `mo_dds`
Dynamically Dimensioned Search (DDS)

Functions/Subroutines

- real(dp) function, dimension(size(pini)), public `mo_dds::dds` (eval, obj_func, pini, prange, r, seed, maxiter, maxit, mask, tmp_file, funcbest, history)
DDS.
- real(dp) function, dimension(size(pini)), public `mo_dds::mdds` (eval, obj_func, pini, prange, seed, maxiter, maxit, mask, tmp_file, funcbest, history)
MDDS.
- subroutine `mo_dds::neigh_value` (x_cur, x_min, x_max, r, new_value)

17.28 mo_errormeasures.f90 File Reference

Data Types

- interface `mo_errormeasures::bias`
- interface `mo_errormeasures::kge`
Kling-Gupta-Efficiency measure.

- interface [mo_errormeasures::kgenocorr](#)
Kling-Gupta-Efficiency measure without correlation.
- interface [mo_errormeasures::lnnse](#)
- interface [mo_errormeasures::mae](#)
- interface [mo_errormeasures::mse](#)
- interface [mo_errormeasures::nse](#)
- interface [mo_errormeasures::sae](#)
- interface [mo_errormeasures::sse](#)
- interface [mo_errormeasures::rmse](#)
- interface [mo_errormeasures::wnse](#)

Modules

- module [mo_errormeasures](#)

Functions/Subroutines

- real(sp) function [mo_errormeasures::bias_sp_1d](#) (x, y, mask)
- real(dp) function [mo_errormeasures::bias_dp_1d](#) (x, y, mask)
- real(sp) function [mo_errormeasures::bias_sp_2d](#) (x, y, mask)
- real(dp) function [mo_errormeasures::bias_dp_2d](#) (x, y, mask)
- real(sp) function [mo_errormeasures::bias_sp_3d](#) (x, y, mask)
- real(dp) function [mo_errormeasures::bias_dp_3d](#) (x, y, mask)
- real(sp) function [mo_errormeasures::kge_sp_1d](#) (x, y, mask)
- real(sp) function [mo_errormeasures::kge_sp_2d](#) (x, y, mask)
- real(sp) function [mo_errormeasures::kge_sp_3d](#) (x, y, mask)
- real(dp) function [mo_errormeasures::kge_dp_1d](#) (x, y, mask)
- real(dp) function [mo_errormeasures::kge_dp_2d](#) (x, y, mask)
- real(dp) function [mo_errormeasures::kge_dp_3d](#) (x, y, mask)
- real(sp) function [mo_errormeasures::kgenocorr_sp_1d](#) (x, y, mask)
- real(sp) function [mo_errormeasures::kgenocorr_sp_2d](#) (x, y, mask)
- real(sp) function [mo_errormeasures::kgenocorr_sp_3d](#) (x, y, mask)
- real(dp) function [mo_errormeasures::kgenocorr_dp_1d](#) (x, y, mask)
- real(dp) function [mo_errormeasures::kgenocorr_dp_2d](#) (x, y, mask)
- real(dp) function [mo_errormeasures::kgenocorr_dp_3d](#) (x, y, mask)
- real(sp) function [mo_errormeasures::lnnse_sp_1d](#) (x, y, mask)
- real(dp) function [mo_errormeasures::lnnse_dp_1d](#) (x, y, mask)
- real(sp) function [mo_errormeasures::lnnse_sp_2d](#) (x, y, mask)
- real(dp) function [mo_errormeasures::lnnse_dp_2d](#) (x, y, mask)
- real(sp) function [mo_errormeasures::lnnse_sp_3d](#) (x, y, mask)
- real(dp) function [mo_errormeasures::lnnse_dp_3d](#) (x, y, mask)
- real(sp) function [mo_errormeasures::mae_sp_1d](#) (x, y, mask)
- real(dp) function [mo_errormeasures::mae_dp_1d](#) (x, y, mask)
- real(sp) function [mo_errormeasures::mae_sp_2d](#) (x, y, mask)
- real(dp) function [mo_errormeasures::mae_dp_2d](#) (x, y, mask)
- real(sp) function [mo_errormeasures::mae_sp_3d](#) (x, y, mask)
- real(dp) function [mo_errormeasures::mae_dp_3d](#) (x, y, mask)
- real(sp) function [mo_errormeasures::mse_sp_1d](#) (x, y, mask)
- real(dp) function [mo_errormeasures::mse_dp_1d](#) (x, y, mask)
- real(sp) function [mo_errormeasures::mse_sp_2d](#) (x, y, mask)
- real(dp) function [mo_errormeasures::mse_dp_2d](#) (x, y, mask)
- real(sp) function [mo_errormeasures::mse_sp_3d](#) (x, y, mask)
- real(dp) function [mo_errormeasures::mse_dp_3d](#) (x, y, mask)

- real(sp) function `mo_errormeasures::nse_sp_1d` (x, y, mask)
- real(dp) function `mo_errormeasures::nse_dp_1d` (x, y, mask)
- real(sp) function `mo_errormeasures::nse_sp_2d` (x, y, mask)
- real(dp) function `mo_errormeasures::nse_dp_2d` (x, y, mask)
- real(sp) function `mo_errormeasures::nse_sp_3d` (x, y, mask)
- real(dp) function `mo_errormeasures::nse_dp_3d` (x, y, mask)
- real(sp) function `mo_errormeasures::sae_sp_1d` (x, y, mask)
- real(dp) function `mo_errormeasures::sae_dp_1d` (x, y, mask)
- real(sp) function `mo_errormeasures::sae_sp_2d` (x, y, mask)
- real(dp) function `mo_errormeasures::sae_dp_2d` (x, y, mask)
- real(sp) function `mo_errormeasures::sae_sp_3d` (x, y, mask)
- real(dp) function `mo_errormeasures::sae_dp_3d` (x, y, mask)
- real(sp) function `mo_errormeasures::sse_sp_1d` (x, y, mask)
- real(dp) function `mo_errormeasures::sse_dp_1d` (x, y, mask)
- real(sp) function `mo_errormeasures::sse_sp_2d` (x, y, mask)
- real(dp) function `mo_errormeasures::sse_dp_2d` (x, y, mask)
- real(sp) function `mo_errormeasures::sse_sp_3d` (x, y, mask)
- real(dp) function `mo_errormeasures::sse_dp_3d` (x, y, mask)
- real(sp) function `mo_errormeasures::rmse_sp_1d` (x, y, mask)
- real(dp) function `mo_errormeasures::rmse_dp_1d` (x, y, mask)
- real(sp) function `mo_errormeasures::rmse_sp_2d` (x, y, mask)
- real(dp) function `mo_errormeasures::rmse_dp_2d` (x, y, mask)
- real(sp) function `mo_errormeasures::rmse_sp_3d` (x, y, mask)
- real(dp) function `mo_errormeasures::rmse_dp_3d` (x, y, mask)
- real(sp) function `mo_errormeasures::wnse_sp_1d` (x, y, mask)
- real(dp) function `mo_errormeasures::wnse_dp_1d` (x, y, mask)
- real(sp) function `mo_errormeasures::wnse_sp_2d` (x, y, mask)
- real(dp) function `mo_errormeasures::wnse_dp_2d` (x, y, mask)
- real(sp) function `mo_errormeasures::wnse_sp_3d` (x, y, mask)
- real(dp) function `mo_errormeasures::wnse_dp_3d` (x, y, mask)

17.29 `mo_file.f90` File Reference

Modules

- module `mo_file`
Provides file names and units for mHM.

Variables

- character(len=*), parameter `mo_file::version` = '5.9'
Current mHM model version.
- character(len=*), parameter `mo_file::version_date` = 'June 2018'
Time of current mHM model version release.
- character(len=*), parameter `mo_file::file_main` = 'mhm_driver.f90'
Driver file.
- character(len=*), parameter `mo_file::file_namelist_mhm` = 'mhm.nml'
Namelist file name.
- integer, parameter `mo_file::unamelist_mhm` = 30
Unit for namelist.
- character(len=*), parameter `mo_file::file_namelist_mhm_param` = 'mhm_parameter.nml'
Parameter namelists file name.

- integer, parameter `mo_file::unamelist_mhm_param` = 31
Unit for namelist.
- character(len=*), parameter `mo_file::file_defoutput` = 'mhm_outputs.nml'
file defining mHM's outputs
- integer, parameter `mo_file::udefoutput` = 67
Unit for file defining mHM's outputs.
- integer, parameter `mo_file::utws` = 77
unit for tws time series

17.30 mo_finish.f90 File Reference

Modules

- module `mo_finish`
Finish a program gracefully.

Functions/Subroutines

- subroutine, public `mo_finish::finish` (name, text, unit)
Finish a program gracefully.

17.31 mo_global_variables.f90 File Reference

Data Types

- type `mo_global_variables::twsstructure`

Modules

- module `mo_global_variables`
Global variables ONLY used in reading, writing and startup.

Variables

- integer(i4) `mo_global_variables::timestep_model_outputs`
- logical, dimension(noutflxstate) `mo_global_variables::outputflxstate`
- integer(i4), dimension(:), allocatable, public `mo_global_variables::timestep_model_inputs`
- logical, public `mo_global_variables::read_meteo_weights`
- character(256), public `mo_global_variables::inputformat_meteo_forcings`
- integer(i4), public `mo_global_variables::timestep_sm_input`
- integer(i4), public `mo_global_variables::timestep_neutrons_input`
- integer(i4), public `mo_global_variables::timestep_et_input`
- character(256), dimension(:), allocatable, public `mo_global_variables::dirprecipitation`
- character(256), dimension(:), allocatable, public `mo_global_variables::dirtemperature`
- character(256), dimension(:), allocatable, public `mo_global_variables::dirmintemperature`
- character(256), dimension(:), allocatable, public `mo_global_variables::dirmaxtemperature`
- character(256), dimension(:), allocatable, public `mo_global_variables::dirnetradiation`
- character(256), dimension(:), allocatable, public `mo_global_variables::dirabsvappressure`
- character(256), dimension(:), allocatable, public `mo_global_variables::dirwindspeed`

- character(256), dimension(:), allocatable, public `mo_global_variables::dirreferenceet`
- character(256), dimension(:), allocatable, public `mo_global_variables::dirsoil_moisture`
- character(256), dimension(:), allocatable, public `mo_global_variables::filetws`
- character(256), dimension(:), allocatable, public `mo_global_variables::dirneutrons`
- character(256), dimension(:), allocatable, public `mo_global_variables::direvapotranspiration`
- integer(i4), parameter, public `mo_global_variables::routingstates` = 2
- type(twsstructure), public `mo_global_variables::basin_avg_tws_obs`
- real(dp), dimension(:, :, :), allocatable, public `mo_global_variables::basin_avg_tws_sim`
- integer(i4), public `mo_global_variables::nmeasperday_tws`
- type(grid), dimension(:), allocatable, public `mo_global_variables::level2`
- real(dp), dimension(:, :, :, :), allocatable, public `mo_global_variables::l1_temp_weights`
- real(dp), dimension(:, :, :, :), allocatable, public `mo_global_variables::l1_pet_weights`
- real(dp), dimension(:, :, :, :), allocatable, public `mo_global_variables::l1_pre_weights`
- real(dp), dimension(:, :, :), allocatable, public `mo_global_variables::l1_pre`
- real(dp), dimension(:, :, :), allocatable, public `mo_global_variables::l1_temp`
- real(dp), dimension(:, :, :), allocatable, public `mo_global_variables::l1_pet`
- real(dp), dimension(:, :, :), allocatable, public `mo_global_variables::l1_tmin`
- real(dp), dimension(:, :, :), allocatable, public `mo_global_variables::l1_tmax`
- real(dp), dimension(:, :, :), allocatable, public `mo_global_variables::l1_netrad`
- real(dp), dimension(:, :, :), allocatable, public `mo_global_variables::l1_absvappress`
- real(dp), dimension(:, :, :), allocatable, public `mo_global_variables::l1_windspeed`
- real(dp), dimension(:, :, :), allocatable, public `mo_global_variables::l1_sm`
- logical, dimension(:, :, :), allocatable, public `mo_global_variables::l1_sm_mask`
- integer(i4) `mo_global_variables::ntimesteps_l1_sm`
- integer(i4) `mo_global_variables::nsoilhorizons_sm_input`
- real(dp), dimension(:, :, :), allocatable, public `mo_global_variables::l1_neutronsdata`
- logical, dimension(:, :, :), allocatable, public `mo_global_variables::l1_neutronsdata_mask`
- integer(i4) `mo_global_variables::ntimesteps_l1_neutrons`
- real(dp), dimension(:, :, :), allocatable, public `mo_global_variables::l1_et`
- logical, dimension(:, :, :), allocatable, public `mo_global_variables::l1_et_mask`
- integer(i4) `mo_global_variables::ntimesteps_l1_et`
- real(dp), dimension(:), allocatable, public `mo_global_variables::l1_inter`
- real(dp), dimension(:), allocatable, public `mo_global_variables::l1_snowpack`
- real(dp), dimension(:), allocatable, public `mo_global_variables::l1_sealstw`
- real(dp), dimension(:, :, :), allocatable, public `mo_global_variables::l1_soilmoist`
- real(dp), dimension(:, :, :), allocatable, public `mo_global_variables::l1_unsatstw`
- real(dp), dimension(:, :, :), allocatable, public `mo_global_variables::l1_satstw`
- real(dp), dimension(:), allocatable, public `mo_global_variables::l1_neutrons`
- real(dp), dimension(:), allocatable, public `mo_global_variables::l1_pet_calc`
- real(dp), dimension(:, :, :), allocatable, public `mo_global_variables::l1_aetsoil`
- real(dp), dimension(:, :, :), allocatable, public `mo_global_variables::l1_aetcanopy`
- real(dp), dimension(:, :, :), allocatable, public `mo_global_variables::l1_aetsealed`
- real(dp), dimension(:, :, :), allocatable, public `mo_global_variables::l1_baseflow`
- real(dp), dimension(:, :, :), allocatable, public `mo_global_variables::l1_infilsoil`
- real(dp), dimension(:, :, :), allocatable, public `mo_global_variables::l1_fastrunoff`
- real(dp), dimension(:, :, :), allocatable, public `mo_global_variables::l1_melt`
- real(dp), dimension(:, :, :), allocatable, public `mo_global_variables::l1_percol`
- real(dp), dimension(:, :, :), allocatable, public `mo_global_variables::l1_preeffect`
- real(dp), dimension(:, :, :), allocatable, public `mo_global_variables::l1_rain`
- real(dp), dimension(:, :, :), allocatable, public `mo_global_variables::l1_runoffseal`
- real(dp), dimension(:, :, :), allocatable, public `mo_global_variables::l1_slowrunoff`
- real(dp), dimension(:, :, :), allocatable, public `mo_global_variables::l1_snow`
- real(dp), dimension(:, :, :), allocatable, public `mo_global_variables::l1_throughfall`
- real(dp), dimension(:, :, :), allocatable, public `mo_global_variables::l1_total_runoff`
- real(dp), dimension(int(yearmonths, i4)), public `mo_global_variables::evap_coeff`

- real(dp), dimension(int(yearmonths, i4)), public `mo_global_variables::fday_prec`
- real(dp), dimension(int(yearmonths, i4)), public `mo_global_variables::fnight_prec`
- real(dp), dimension(int(yearmonths, i4)), public `mo_global_variables::fday_pet`
- real(dp), dimension(int(yearmonths, i4)), public `mo_global_variables::fnight_pet`
- real(dp), dimension(int(yearmonths, i4)), public `mo_global_variables::fday_temp`
- real(dp), dimension(int(yearmonths, i4)), public `mo_global_variables::fnight_temp`
- real(dp), dimension(:), allocatable, public `mo_global_variables::neutron_integral_afast`

17.32 mo_grid.f90 File Reference

Modules

- module `mo_grid`

TODO: add description.

Functions/Subroutines

- subroutine, public `mo_grid::init_lowres_level` (highres, target_resolution, lowres, highres_lowres_remap)
Level-1 variable initialization.
- subroutine, public `mo_grid::set_basin_indices` (grids)
TODO: add description.
- subroutine, public `mo_grid::l0_grid_setup` (new_grid)
level 0 variable initialization
- subroutine, public `mo_grid::mapcoordinates` (level, y, x)
Generate map coordinates.
- subroutine, public `mo_grid::geocoordinates` (level, lat, lon)
Generate geographic coordinates.
- subroutine `mo_grid::calculate_grid_properties` (nrowsIn, ncolsIn, xllcornerIn, yllcornerIn, cellsizeln, aiming← Resolution, nrowsOut, ncolsOut, xllcornerOut, yllcornerOut, cellsizeOut)
Calculates basic grid properties at a required coarser level using information of a given finer level. Calculates basic grid properties at a required coarser level (e.g., L11) using information of a given finer level (e.g., L0). Basic grid properties such as nrows, ncols, xllcorner, yllcorner, cellsize are estimated in this routine.

17.33 mo_init_states.f90 File Reference

Modules

- module `mo_init_states`

Initialization of all state variables of mHM.

Functions/Subroutines

- subroutine, public `mo_init_states::variables_alloc` (ncells1)
Allocation of space for mHM related L1 and L11 variables.
- subroutine, public `mo_init_states::variables_default_init`
Default initialization mHM related L1 variables.

17.34 mo_julian.f90 File Reference

Data Types

- interface [mo_julian::setcalendar](#)

Modules

- module [mo_julian](#)
Julian date conversion routines.

Functions/Subroutines

- subroutine [mo_julian::setcalendarstring](#) (selector)
Set module private variable calendar.
- subroutine [mo_julian::setcalendarinteger](#) (selector)
Set module private variable calendar.
- pure integer(i4) function [mo_julian::selectcalendar](#) (selector)
Select a calendar.
- elemental subroutine, public [mo_julian::caldat](#) (julian, dd, mm, yy, calendar)
Day, month and year from Julian day in the current or given calendar.
- elemental subroutine, public [mo_julian::dec2date](#) (julian, dd, mm, yy, hh, nn, ss, calendar)
Day, month, year, hour, minute, and second from fractional Julian day in the current or given calendar.
- elemental real(dp) function, public [mo_julian::date2dec](#) (dd, mm, yy, hh, nn, ss, calendar)
Fractional Julian day from day, month, year, hour, minute, second in the current calendar.
- elemental integer(i4) function, public [mo_julian::julday](#) (dd, mm, yy, calendar)
Julian day from day, month and year in the current or given calendar.
- elemental subroutine, public [mo_julian::caldatjulian](#) (julian, dd, mm, yy)
Day, month and year from Julian day.
- elemental real(dp) function [mo_julian::date2decjulian](#) (dd, mm, yy, hh, nn, ss)
Fractional Julian day from day, month, year, hour, minute, second.
- elemental subroutine [mo_julian::dec2datejulian](#) (julian, dd, mm, yy, hh, nn, ss)
Day, month, year, hour, minute, and second from fractional Julian day.
- elemental integer(i4) function [mo_julian::juldayjulian](#) (dd, mm, yy)
Julian day from day, month and year.
- elemental integer(i4) function, public [mo_julian::ndays](#) (dd, mm, yy)
IMSL Julian day from day, month and year.
- elemental subroutine, public [mo_julian::ndyin](#) (julian, dd, mm, yy)
Day, month and year from IMSL Julian day.
- elemental subroutine [mo_julian::caldat360](#) (julian, dd, mm, yy)
Day, month and year from Julian day in a 360 day calendar.
- elemental integer(i4) function [mo_julian::julday360](#) (dd, mm, yy)
Julian day from day, month and year in a 360_day calendar.
- elemental subroutine [mo_julian::dec2date360](#) (julian, dd, mm, yy, hh, nn, ss)
Day, month, year, hour, minute, and second from fractional Julian day in a 360_day calendar.
- elemental real(dp) function [mo_julian::date2dec360](#) (dd, mm, yy, hh, nn, ss)
Fractional Julian day from day, month, year, hour, minute, second in 360 day calendar.
- elemental subroutine [mo_julian::caldat365](#) (julian, dd, mm, yy)
Day, month and year from Julian day in a 365 day calendar.
- elemental integer(i4) function [mo_julian::julday365](#) (dd, mm, yy)

Julian day from day, month and year in a 365_day calendar.

- elemental subroutine `mo Julian::dec2date365` (julian, dd, mm, yy, hh, nn, ss)

Day, month, year, hour, minute, and second from fractional Julian day in a 365_day calendar.

- elemental real(dp) function `mo Julian::date2dec365` (dd, mm, yy, hh, nn, ss)

Fractional Julian day from day, month, year, hour, minute, second in 365 day calendar.

Variables

- integer(i4), save, private `mo Julian::calendar` = 1

17.35 mo_kind.f90 File Reference

Data Types

- type `mo Kind::sprs2_sp`

Single Precision Numerical Recipes types for sparse arrays.

- type `mo Kind::sprs2_dp`

Double Precision Numerical Recipes types for sparse arrays.

Modules

- module `mo kind`

Define number representations.

Variables

- integer, parameter `mo kind::i1` = SELECTED_INT_KIND(2)

1 Byte Integer Kind

- integer, parameter `mo kind::i2` = c_short

2 Byte Integer Kind

- integer, parameter `mo kind::i4` = c_int

4 Byte Integer Kind

- integer, parameter `mo kind::i8` = c_long_long

8 Byte Integer Kind

- integer, parameter `mo kind::sp` = c_float

Single Precision Real Kind.

- integer, parameter `mo kind::dp` = c_double

Double Precision Real Kind.

- integer, parameter `mo kind::spc` = c_float_complex

Single Precision Complex Kind.

- integer, parameter `mo kind::dpc` = c_double_complex

Double Precision Complex Kind.

- integer, parameter `mo kind::lgt` = KIND(.true.)

Logical Kind.

17.36 mo_linfit.f90 File Reference

Data Types

- interface [mo_linfit::linfit](#)

Fits a straight line to input data by minimizing χ^2 .

Modules

- module [mo_linfit](#)

Fitting a straight line.

Functions/Subroutines

- real(dp) function, dimension(:), allocatable [mo_linfit::linfit_dp](#) (x, y, a, b, siga, sigb, chi2, model2)
- real(sp) function, dimension(:), allocatable [mo_linfit::linfit_sp](#) (x, y, a, b, siga, sigb, chi2, model2)

17.37 mo_mcmc.f90 File Reference

Data Types

- interface [mo_mcmc::mcmc](#)

This module is Monte Carlo Markov Chain sampling of a posterior parameter distribution (measurement errors are either known or modeled).

- interface [mo_mcmc::mcmc_stddev](#)

This module is Monte Carlo Markov Chain sampling of a posterior parameter distribution (measurement errors are neither known nor modeled).

Modules

- module [mo_mcmc](#)

This module is Monte Carlo Markov Chain sampling of a posterior parameter distribution.

Functions/Subroutines

- subroutine [mo_mcmc::mcmc_dp](#) (eval, likelihood, para, rangePar, mcmc_paras, burnin_paras, seed_in, printflag_in, maskpara_in, restart, restart_file, tmp_file, loglike_in, ParaSelectMode_in, iter_burnin_in, iter_mcmc_in, chains_in, stepsize_in)
- subroutine [mo_mcmc::mcmc_stddev_dp](#) (eval, likelihood, para, rangePar, mcmc_paras, burnin_paras, seed_in, printflag_in, maskpara_in, tmp_file, loglike_in, ParaSelectMode_in, iter_burnin_in, iter_mcmc_in, chains_in, stepsize_in)
- real(dp) function [mo_mcmc::pargen_dp](#) (old, dMax, oMin, oMax, RN, inbound)
- real(dp) function [mo_mcmc::pargennorm_dp](#) (old, dMax, oMin, oMax, RN, inbound)
- recursive subroutine [mo_mcmc::generatenewparameterset_dp](#) (ParaSelectMode, paraold, truepara, range←Par, stepsize, save_state_2, save_state_3, paranew, ChangePara)

17.38 mo_message.f90 File Reference

Modules

- module [mo_message](#)
Write out concatenated strings.

Functions/Subroutines

- subroutine, public [mo_message::message](#) (t01, t02, t03, t04, t05, t06, t07, t08, t09, t10, uni, advance)
Write out several string concatenated either on screen or in a file.

Variables

- character(len=1024), public [mo_message::message_text](#) = "

17.39 mo_meteo_forcings.f90 File Reference

Modules

- module [mo_meteo_forcings](#)
Prepare meteorological forcings data for mHM.

Functions/Subroutines

- subroutine, public [mo_meteo_forcings::prepare_meteo_forcings_data](#) (iBasin, tt)
Prepare meteorological forcings data for a given variable.
- subroutine [mo_meteo_forcings::meteo_forcings_wrapper](#) (iBasin, dataPath, inputFormat, dataOut1, lower, upper, ncvarName)
Prepare meteorological forcings data for mHM at Level-1.
- subroutine [mo_meteo_forcings::meteo_weights_wrapper](#) (iBasin, read_meteo_weights, dataPath, dataOut1, lower, upper, ncvarName)
Prepare weights for meteorological forcings data for mHM at Level-1.
- subroutine [mo_meteo_forcings::chunk_config](#) (iBasin, tt, read_flag, readPer)
determines the start date, end date, and read_flag given basin id and current timestep
- logical function [mo_meteo_forcings::is_read](#) (iBasin, tt)
evaluate whether new chunk should be read at this timestep
- subroutine [mo_meteo_forcings::chunk_size](#) (iBasin, tt, readPer)
calculate beginning and end of read Period, i.e. that is length of current chunk to read

17.40 mo_mhm.f90 File Reference

Modules

- module [mo_mhm](#)
Call all main processes of mHM.

Functions/Subroutines

- subroutine, public `mo_mhm::mhm` (read_states, tt, time, processMatrix, horizon_depth, nCells1, nHorizons←_mHM, ntimesteps_day, 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, petLAlcorFactorL1, HarSamCoeff, PrieTayAlpha, aeroResist, surfResist, frac_roots, interc_max, karst←_loss, k0, k1, k2, kp, soil_moist_FC, soil_moist_sat, soil_moist_exponen, jarvis_thresh_c1, temp_thresh, unsat_thresh, water_thresh_sealed, wilting_point)

Pure mHM calculations.

17.41 `mo_mhm_constants.f90` File Reference

Modules

- module `mo_mhm_constants`

Provides mHM specific constants.

Variables

- real(dp), parameter, public `mo_mhm_constants::h2odens` = 1000.0_dp
- real(dp), parameter, public `mo_mhm_constants::p2_initstatefluxes` = 15.00_dp
- real(dp), parameter, public `mo_mhm_constants::p3_initstatefluxes` = 10.00_dp
- real(dp), parameter, public `mo_mhm_constants::p4_initstatefluxes` = 75.00_dp
- real(dp), parameter, public `mo_mhm_constants::p5_initstatefluxes` = 1500.00_dp
- real(dp), parameter, public `mo_mhm_constants::c1_initstatesm` = 0.25_dp
- integer(i4), parameter, public `mo_mhm_constants::noutflxstate` = 20_i4
- real(dp), parameter, public `mo_mhm_constants::stboltzmann` = 5.67e-08_dp

Stefan-Boltzmann constant [W m^-2 K^-4].

- real(dp), parameter, public `mo_mhm_constants::harsamconst` = 17.800_dp

Hargreaves-Samani ref. ET formula [deg C].

- real(dp), parameter, public `mo_mhm_constants::duffiedr` = 0.0330_dp
- real(dp), parameter, public `mo_mhm_constants::duffiedelta1` = 0.4090_dp
- real(dp), parameter, public `mo_mhm_constants::duffiedelta2` = 1.3900_dp
- real(dp), parameter, public `mo_mhm_constants::tetens_c1` = 0.6108_dp

Tetens's formula to calculate saturated vapour pressure.

- real(dp), parameter, public `mo_mhm_constants::tetens_c2` = 17.270_dp
- real(dp), parameter, public `mo_mhm_constants::tetens_c3` = 237.30_dp
- real(dp), parameter, public `mo_mhm_constants::satpressureslope1` = 4098.0_dp

calculation of the slope of the saturation vapour pressure curve following Tetens

- real(dp), parameter, public `mo_mhm_constants::desilets_a0` = 0.0808_dp

Neutrons and moisture: N0 formula, Desilets et al. 2010.

- real(dp), parameter, public `mo_mhm_constants::desilets_a1` = 0.372_dp
- real(dp), parameter, public `mo_mhm_constants::desilets_a2` = 0.115_dp
- real(dp), parameter, public `mo_mhm_constants::cosmic_bd` = 1.4020_dp

Neutrons and moisture: COSMIC, Shuttleworth et al. 2013.

- real(dp), parameter, public `mo_mhm_constants::cosmic_vwclat` = 0.0753_dp
- real(dp), parameter, public `mo_mhm_constants::cosmic_n` = 348.33_dp
- real(dp), parameter, public `mo_mhm_constants::cosmic_alpha` = 0.2392421548_dp

- real(dp), parameter, public `mo_mhm_constants::cosmic_l1` = 161.98621864_dp
- real(dp), parameter, public `mo_mhm_constants::cosmic_l2` = 129.14558985_dp
- real(dp), parameter, public `mo_mhm_constants::cosmic_l3` = 107.82204562_dp
- real(dp), parameter, public `mo_mhm_constants::cosmic_l4` = 3.1627190566_dp

17.42 mo_mhm_eval.f90 File Reference

Modules

- module `mo_mhm_eval`
Runs mhm with a specific parameter set and returns required variables, e.g. runoff.

Functions/Subroutines

- subroutine, public `mo_mhm_eval::mhm_eval` (parameterset, runoff, sm_opti, basin_avg_tws, neutrons_opti, et_opti)
Runs mhm with a specific parameter set and returns required variables, e.g. runoff.

17.43 mo_mhm_read_config.f90 File Reference

Modules

- module `mo_mhm_read_config`
Reading of main model configurations.

Functions/Subroutines

- subroutine, public `mo_mhm_read_config::mhm_read_config` (file_namelist, unamelist)
Read main configurations for mHM.

17.44 mo_moment.f90 File Reference

Data Types

- interface `mo_moment::absdev`
- interface `mo_moment::average`
- interface `mo_moment::central_moment`
- interface `mo_moment::central_moment_var`
- interface `mo_moment::correlation`
- interface `mo_moment::covariance`
- interface `mo_moment::kurtosis`
- interface `mo_moment::mean`
- interface `mo_moment::mixed_central_moment`
- interface `mo_moment::mixed_central_moment_var`
- interface `mo_moment::moment`
- interface `mo_moment::skewness`
- interface `mo_moment::stddev`
- interface `mo_moment::variance`

Modules

- module `mo_moment`

Functions/Subroutines

- real(dp) function `mo_moment::absdev_dp` (dat, mask)
- real(sp) function `mo_moment::absdev_sp` (dat, mask)
- real(dp) function `mo_moment::average_dp` (dat, mask)
- real(sp) function `mo_moment::average_sp` (dat, mask)
- real(dp) function `mo_moment::central_moment_dp` (x, r, mask)
- real(sp) function `mo_moment::central_moment_sp` (x, r, mask)
- real(dp) function `mo_moment::central_moment_var_dp` (x, r, mask)
- real(sp) function `mo_moment::central_moment_var_sp` (x, r, mask)
- real(dp) function `mo_moment::correlation_dp` (x, y, mask)
- real(sp) function `mo_moment::correlation_sp` (x, y, mask)
- real(dp) function `mo_moment::covariance_dp` (x, y, mask)
- real(sp) function `mo_moment::covariance_sp` (x, y, mask)
- real(dp) function `mo_moment::kurtosis_dp` (dat, mask)
- real(sp) function `mo_moment::kurtosis_sp` (dat, mask)
- real(dp) function `mo_moment::mean_dp` (dat, mask)
- real(sp) function `mo_moment::mean_sp` (dat, mask)
- real(dp) function `mo_moment::mixed_central_moment_dp` (x, y, r, s, mask)
- real(sp) function `mo_moment::mixed_central_moment_sp` (x, y, r, s, mask)
- real(dp) function `mo_moment::mixed_central_moment_var_dp` (x, y, r, s, mask)
- real(sp) function `mo_moment::mixed_central_moment_var_sp` (x, y, r, s, mask)
- subroutine `mo_moment::moment_dp` (dat, average, variance, skewness, kurtosis, mean, stddev, absdev, mask)
- subroutine `mo_moment::moment_sp` (dat, average, variance, skewness, kurtosis, mean, stddev, absdev, mask)
- real(dp) function `mo_moment::stddev_dp` (dat, mask)
- real(sp) function `mo_moment::stddev_sp` (dat, mask)
- real(dp) function `mo_moment::skewness_dp` (dat, mask)
- real(sp) function `mo_moment::skewness_sp` (dat, mask)
- real(dp) function `mo_moment::variance_dp` (dat, mask)
- real(sp) function `mo_moment::variance_sp` (dat, mask)

17.45 `mo_mpr_constants.f90` File Reference

Modules

- module `mo_mpr_constants`

Provides MPR specific constants.

Variables

- integer(i4), parameter, public `mo_mpr_constants::nlcover_class` = 3_i4
- integer(i4), parameter, public `mo_mpr_constants::maxgeounit` = 25_i4
- integer(i4), parameter, public `mo_mpr_constants::maxnosoilhorizons` = 10_i4
- real(dp), parameter, public `mo_mpr_constants::p2_initstatefluxes` = 15.00_dp
- real(dp), parameter, public `mo_mpr_constants::p3_initstatefluxes` = 10.00_dp
- real(dp), parameter, public `mo_mpr_constants::p4_initstatefluxes` = 75.00_dp

- real(dp), parameter, public `mo_mpr_constants::p5_initstatefluxes` = 1500.00_dp
 - real(dp), parameter, public `mo_mpr_constants::c1_initstatesm` = 0.25_dp
 - real(dp), parameter, public `mo_mpr_constants::bulkdens_orgmatter` = 0.224_dp
 - real(dp), parameter, public `mo_mpr_constants::field_cap_c1` = -0.60_dp
 - real(dp), parameter, public `mo_mpr_constants::field_cap_c2` = 2.0_dp
 - real(dp), parameter, public `mo_mpr_constants::vgenuchten_sandtresh` = 66.5_dp
 - real(dp), parameter, public `mo_mpr_constants::vgenuchtenn_c1` = 1.392_dp
 - real(dp), parameter, public `mo_mpr_constants::vgenuchtenn_c2` = 0.418_dp
 - real(dp), parameter, public `mo_mpr_constants::vgenuchtenn_c3` = -0.024_dp
 - real(dp), parameter, public `mo_mpr_constants::vgenuchtenn_c4` = 1.212_dp
 - real(dp), parameter, public `mo_mpr_constants::vgenuchtenn_c5` = -0.704_dp
 - real(dp), parameter, public `mo_mpr_constants::vgenuchtenn_c6` = -0.648_dp
 - real(dp), parameter, public `mo_mpr_constants::vgenuchtenn_c7` = 0.023_dp
 - real(dp), parameter, public `mo_mpr_constants::vgenuchtenn_c8` = 0.044_dp
 - real(dp), parameter, public `mo_mpr_constants::vgenuchtenn_c9` = 3.168_dp
 - real(dp), parameter, public `mo_mpr_constants::vgenuchtenn_c10` = -2.562_dp
 - real(dp), parameter, public `mo_mpr_constants::vgenuchtenn_c11` = 7.0E-9_dp
 - real(dp), parameter, public `mo_mpr_constants::vgenuchtenn_c12` = 4.004_dp
 - real(dp), parameter, public `mo_mpr_constants::vgenuchtenn_c13` = 3.750_dp
 - real(dp), parameter, public `mo_mpr_constants::vgenuchtenn_c14` = -0.016_dp
 - real(dp), parameter, public `mo_mpr_constants::vgenuchtenn_c15` = -4.197_dp
 - real(dp), parameter, public `mo_mpr_constants::vgenuchtenn_c16` = 0.013_dp
 - real(dp), parameter, public `mo_mpr_constants::vgenuchtenn_c17` = 0.076_dp
 - real(dp), parameter, public `mo_mpr_constants::vgenuchtenn_c18` = 0.276_dp
 - real(dp), parameter, public `mo_mpr_constants::ks_c` = 10.0_dp
 - real(dp), parameter, public `mo_mpr_constants::pwp_c` = 1.0_dp
 - real(dp), parameter, public `mo_mpr_constants::pwp_matpot_theta` = 15000.0_dp
 - real(dp), parameter, public `mo_mpr_constants::windmeasheight` = 10.0_dp
- assumed meteorol. measurement hight for estimation of aeroResist and surfResist*
- real(dp), parameter, public `mo_mpr_constants::karman` = 0.41_dp
- von Karman constant*
- real(dp), parameter, public `mo_mpr_constants::lai_factor_surfresi` = 0.3_dp
- LAI factor for bulk surface resistance formulation.*
- real(dp), parameter, public `mo_mpr_constants::lai_offset_surfresi` = 1.2_dp
- LAI offset for bulk surface resistance formulation.*
- real(dp), parameter, public `mo_mpr_constants::max_surfresist` = 250.0_dp
- maximum bulk surface resistance*

17.46 mo_mpr_eval.f90 File Reference

Modules

- module `mo_mpr_eval`

Runs MPR and writes to global effective parameters.

Functions/Subroutines

- subroutine, public `mo_mpr_eval::mpr_eval` (parameterset)

Runs MPR and writes to global effective parameters.

17.47 mo_mpr_file.f90 File Reference

Modules

- module **mo_mpr_file**
Provides file names and units for mRM.

Variables

- character(len=*), parameter **mo_mpr_file::version** = '0.1'
Current mHM model version.
- character(len=*), parameter **mo_mpr_file::version_date** = 'Jun 2018'
Time of current mHM model version release.
- character(len=*), parameter **mo_mpr_file::file_main** = 'mpr_driver.f90'
Driver file.
- character(len=*), parameter **mo_mpr_file::file_namelist_mpr** = 'mpr.nml'
Namelist file name.
- integer, parameter **mo_mpr_file::unamelist_mpr** = 80
Unit for namelist.
- character(len=*), parameter **mo_mpr_file::file_namelist_mpr_param** = 'mpr_parameter.nml'
Parameter namelists file name.
- integer, parameter **mo_mpr_file::unamelist_mpr_param** = 31
Unit for namelist.
- character(len=*), parameter **mo_mpr_file::file_soil_database** = 'soil_classdefinition.txt'
Soil database file (iFlag_soilDB = 0) = classical mHM format.
- character(len=*), parameter **mo_mpr_file::file_soil_database_1** = 'soil_classdefinition_iFlag_soilDB_1.txt'
Soil database file (iFlag_soilDB = 1)
- integer, parameter **mo_mpr_file::usoil_database** = 52
Unit for soil data base.
- character(len=*), parameter **mo_mpr_file::file_slope** = 'slope.asc'
slope input data file
- integer, parameter **mo_mpr_file::uslope** = 54
Unit for slope input data file.
- character(len=*), parameter **mo_mpr_file::file_aspect** = 'aspect.asc'
aspect input data file
- integer, parameter **mo_mpr_file::uaspect** = 55
Unit for aspect input data file.
- character(len=*), parameter **mo_mpr_file::file_hydrogeoclass** = 'geology_class.asc'
hydrogeological classes input data file
- integer, parameter **mo_mpr_file::uhydrogeoclass** = 58
Unit for hydrogeological classes input data file.
- character(len=*), parameter **mo_mpr_file::file_soilclass** = 'soil_class.asc'
soil classes input data file
- integer, parameter **mo_mpr_file::usoilclass** = 59
Unit for soil classes input data file.
- character(len=*), parameter **mo_mpr_file::file_laiclass** = 'LAI_class.asc'
LAI classes input data file.
- integer, parameter **mo_mpr_file::ulaiclass** = 60
Unit for LAI input data file.
- character(len=*), parameter **mo_mpr_file::file_geolut** = 'geology_classdefinition.txt'

- *geological formation lookup table file*
- integer, parameter `mo_mpr_file::ugeolut` = 64
 - Unit for geological formation lookup table file.*
- character(len=*), parameter `mo_mpr_file::file_lailut` = 'LAI_classdefinition.txt'
 - LAI classes lookup table file.*
- integer, parameter `mo_mpr_file::ulailut` = 65
 - Unit for LAI classes lookup table file.*
- character(len=*), parameter `mo_mpr_file::file_meteo_header` = 'header.txt'
 - Input nCols and nRows of binary meteo and LAI files are in header file.*
- integer, parameter `mo_mpr_file::umeteo_header` = 50
 - Unit for meteo header file.*
- character(len=*), parameter `mo_mpr_file::file_meteo_binary_end` = '.bin'
 - File ending of meteo files.*
- integer, parameter `mo_mpr_file::umeteo` = 51
 - Unit for meteo files.*

17.48 mo_mpr_global_variables.f90 File Reference

Data Types

- type `mo_mpr_global_variables::soiltype`

Modules

- module `mo_mpr_global_variables`
 - Global variables for mpr only.*

Variables

- real(dp), public `mo_mpr_global_variables::c2tstu`
- real(dp), public `mo_mpr_global_variables::tillagedepth`
- integer(i4), public `mo_mpr_global_variables::nsoiltypes`
- integer(i4), public `mo_mpr_global_variables::iflag_soildb`
- integer(i4), public `mo_mpr_global_variables::nsoilhorizons_mhm`
- real(dp), dimension(:), allocatable, public `mo_mpr_global_variables::horizondepth_mhm`
- type(soiltype), public `mo_mpr_global_variables::soildb`
- integer(i4), public `mo_mpr_global_variables::ngeounits`
- integer(i4), dimension(:), allocatable, public `mo_mpr_global_variables::geounitlist`
- integer(i4), dimension(:), allocatable, public `mo_mpr_global_variables::geounitkar`
- character(256), public `mo_mpr_global_variables::inputformat_gridded_lai`
- integer(i4), public `mo_mpr_global_variables::timestep_lai_input`
- integer(i4), public `mo_mpr_global_variables::nlaiiclass`
- integer(i4), public `mo_mpr_global_variables::nlai`
- integer(i4), dimension(:), allocatable, public `mo_mpr_global_variables::laiunitlist`
- real(dp), dimension(:, :), allocatable, public `mo_mpr_global_variables::lailut`
- type(period), dimension(:, :), allocatable, public `mo_mpr_global_variables::laiper`
- real(dp), public `mo_mpr_global_variables::fracsealed_cityarea`
- real(dp), dimension(:, :), allocatable, public `mo_mpr_global_variables::l0_slope_emp`
- real(dp), dimension(:, :), allocatable, public `mo_mpr_global_variables::l0_gridded_lai`
- real(dp), dimension(:, :), allocatable, public `mo_mpr_global_variables::l0_slope`
- real(dp), dimension(:, :), allocatable, public `mo_mpr_global_variables::l0_asp`

- integer(i4), dimension(:, :, :), allocatable, public `mo_mpr_global_variables::l0_soilid`
- integer(i4), dimension(:, :, :), allocatable, public `mo_mpr_global_variables::l0_geounit`
- character(256), dimension(:, :, :), allocatable, public `mo_mpr_global_variables::dirgridded_lai`
- real(dp), dimension(:, :, :, :), allocatable, public `mo_mpr_global_variables::l1_fsealed`
- real(dp), dimension(:, :, :, :), allocatable, public `mo_mpr_global_variables::l1_alpha`
- real(dp), dimension(:, :, :, :), allocatable, public `mo_mpr_global_variables::l1_deggdayinc`
- real(dp), dimension(:, :, :, :), allocatable, public `mo_mpr_global_variables::l1_deggdaymax`
- real(dp), dimension(:, :, :, :), allocatable, public `mo_mpr_global_variables::l1_deggdaynopre`
- real(dp), dimension(:, :, :, :), allocatable, public `mo_mpr_global_variables::l1_deggday`
- real(dp), dimension(:, :, :, :), allocatable, public `mo_mpr_global_variables::l1_karstloss`
- real(dp), dimension(:, :, :, :), allocatable, public `mo_mpr_global_variables::l1_fasp`
- real(dp), dimension(:, :, :, :), allocatable, public `mo_mpr_global_variables::l1_petlaicorfactor`
- real(dp), dimension(:, :, :, :), allocatable, public `mo_mpr_global_variables::l1_harsamcoeff`
- real(dp), dimension(:, :, :, :), allocatable, public `mo_mpr_global_variables::l1_prietaryalpha`
- real(dp), dimension(:, :, :, :), allocatable, public `mo_mpr_global_variables::l1_aeroresist`
- real(dp), dimension(:, :, :, :), allocatable, public `mo_mpr_global_variables::l1_surfrresist`
- real(dp), dimension(:, :, :, :), allocatable, public `mo_mpr_global_variables::l1_froots`
- real(dp), dimension(:, :, :, :), allocatable, public `mo_mpr_global_variables::l1_maxinter`
- real(dp), dimension(:, :, :, :), allocatable, public `mo_mpr_global_variables::l1_kfastflow`
- real(dp), dimension(:, :, :, :), allocatable, public `mo_mpr_global_variables::l1_kslowflow`
- real(dp), dimension(:, :, :, :), allocatable, public `mo_mpr_global_variables::l1_kbaseflow`
- real(dp), dimension(:, :, :, :), allocatable, public `mo_mpr_global_variables::l1_kperco`
- real(dp), dimension(:, :, :, :), allocatable, public `mo_mpr_global_variables::l1_soilmoistfc`
- real(dp), dimension(:, :, :, :), allocatable, public `mo_mpr_global_variables::l1_soilmoistsat`
- real(dp), dimension(:, :, :, :), allocatable, public `mo_mpr_global_variables::l1_soilmoistexp`
- real(dp), dimension(:, :, :, :), allocatable, public `mo_mpr_global_variables::l1_jarvis_thresh_c1`
- real(dp), dimension(:, :, :, :), allocatable, public `mo_mpr_global_variables::l1_tempthresh`
- real(dp), dimension(:, :, :, :), allocatable, public `mo_mpr_global_variables::l1_unsatthresh`
- real(dp), dimension(:, :, :, :), allocatable, public `mo_mpr_global_variables::l1_sealedthresh`
- real(dp), dimension(:, :, :, :), allocatable, public `mo_mpr_global_variables::l1_wiltingpoint`

17.49 `mo_mpr_pet.f90` File Reference

Modules

- module `mo_mpr_pet`

TODO: add description.

Functions/Subroutines

- subroutine, public `mo_mpr_pet::pet_correctbylai` (param, nodata, LCOVER0, LAI0, mask0, cell_id0, upp_<row_L1, low_row_L1, lef_col_L1, rig_col_L1, nL0_in_L1, L1_petLAcorFactor)

estimate PET correction factor based on LAI at L1
- subroutine, public `mo_mpr_pet::pet_correctbyasp` (Id0, latitude_l0, Asp0, param, nodata, fAsp0)

correction of PET
- subroutine, public `mo_mpr_pet::priestley_taylor_alpha` (LAI0, param, mask0, nodata, cell_id0, nL0_in_L1, Upp_row_L1, Low_row_L1, Lef_col_L1, Rig_col_L1, priestley_taylor_alpha1)

Regionalization of priestley taylor alpha.
- subroutine, public `mo_mpr_pet::bulksurface_resistance` (LAI0, param, mask0, nodata, cell_id0, nL0_in_L1, Upp_row_L1, Low_row_L1, Lef_col_L1, Rig_col_L1, bulksurface_resistance1)

Regionalization of bulk surface resistance.

17.50 mo_mpr_read_config.f90 File Reference

Modules

- module [mo_mpr_read_config](#)
read mpr config

Functions/Subroutines

- subroutine, public [mo_mpr_read_config::mpr_read_config](#) (file_namelist, unamelist, file_namelist_param, unamelist_param)
Read the general config of mpr.

17.51 mo_mpr_restart.f90 File Reference

Data Types

- interface [mo_mpr_restart::unpack_field_and_write](#)
TODO: add description.

Modules

- module [mo_mpr_restart](#)
reading and writing states, fluxes and configuration for restart of mHM.

Functions/Subroutines

- subroutine, public [mo_mpr_restart::write_mpr_restart_files](#) (OutPath)
write restart files for each basin
- subroutine, public [mo_mpr_restart::write_eff_params](#) (mask1, s1, e1, rows1, cols1, soil1, lcscenes, lais, nc)
TODO: add description.
- subroutine [mo_mpr_restart::unpack_field_and_write_1d_i4](#) (nc, var_name, var_dims, fill_value, data, mask, var_long_name)
- subroutine [mo_mpr_restart::unpack_field_and_write_1d_dp](#) (nc, var_name, var_dims, fill_value, data, mask, var_long_name)
- subroutine [mo_mpr_restart::unpack_field_and_write_2d_dp](#) (nc, var_name, var_dims, fill_value, data, mask, var_long_name)
- subroutine [mo_mpr_restart::unpack_field_and_write_3d_dp](#) (nc, var_name, var_dims, fill_value, data, mask, var_long_name)

17.52 mo_mpr_runoff.f90 File Reference

Modules

- module [mo_mpr_runoff](#)
multiscale parameter regionalization for runoff generation

Functions/Subroutines

- subroutine, public [mo_mpr_runoff::mpr_runoff](#) (LCOVER0, mask0, SMs_FC0, slope_emp0, KsVar_H0, param, cell_id0, upp_row_L1, low_row_L1, lef_col_L1, rig_col_L1, nL0_in_L1, c2TSTu, L1_HL1, L1_K0, L1_K1, L1_alpha)
multiscale parameter regionalization for runoff parameters

17.53 mo_mpr_smhorizons.f90 File Reference

Modules

- module [mo_mpr_smhorizons](#)
setting up the soil moisture horizons

Functions/Subroutines

- subroutine, public [mo_mpr_smhorizons::mpr_smhorizons](#) (param, processMatrix, iFlag_soil, nHorizons_mHM, HorizonDepth, LCOVER0, soilID0, nHorizons, nTillHorizons, thetaS_till, thetaFC_till, thetaPW_till, thetaS, thetaFC, thetaPW, Wd, Db, DbM, RZdepth, mask0, cell_id0, upp_row_L1, low_row_L1, lef_col_L1, rig_col_L1, nL0_in_L1, L1_beta, L1_SMs, L1_FC, L1_PW, L1_fRoots)
upscale soil moisture horizons

17.54 mo_mpr_soilmoist.f90 File Reference

Modules

- module [mo_mpr_soilmoist](#)
Multiscale parameter regionalization (MPR) for soil moisture.

Functions/Subroutines

- subroutine, public [mo_mpr_soilmoist::mpr_sm](#) (param, is_present, nHorizons, nTillHorizons, sand, clay, DbM, ID0, soilID0, LCover0, thetaS_till, thetaFC_till, thetaPW_till, thetaS, thetaFC, thetaPW, Ks, Db, KsVar_H0, KsVar_V0, SMs_FC0)
multiscale parameter regionalization for soil moisture
- elemental pure subroutine [mo_mpr_soilmoist::pwp](#) (Genu_Mual_n, Genu_Mual_alpha, thetaS, thetaPWP)
Permanent Wilting point.
- elemental pure subroutine [mo_mpr_soilmoist::field_cap](#) (thetaFC, Ks, thetaS, Genu_Mual_n)
calculates the field capacity
- subroutine [mo_mpr_soilmoist::genuchten](#) (thetaS, Genu_Mual_n, Genu_Mual_alpha, param, sand, clay, Db)
calculates the Genuchten shape parameter
- subroutine [mo_mpr_soilmoist::hydro_cond](#) (KS, param, sand, clay)
calculates the hydraulic conductivity KS

17.55 mo_mpr_startup.f90 File Reference

Modules

- module [mo_mpr_startup](#)
Startup procedures for mHM.

Functions/Subroutines

- subroutine, public `mo_mpr_startup::mpr_initialize`
Initialize main mHM variables.
- subroutine `mo_mpr_startup::l0_check_input` (iBasin)
Check for errors in L0 input data.
- subroutine `mo_mpr_startup::l0_variable_init` (iBasin)
level 0 variable initialization
- subroutine, public `mo_mpr_startup::init_eff_params` (ncells1)
Allocation of space for mHM related L1 and L11 variables.

17.56 mo_mrm_constants.f90 File Reference

Modules

- module `mo_mrm_constants`
Provides mRM specific constants.

Variables

- integer(i4), parameter, public `mo_mrm_constants::noutfluxstate` = 1_i4
- integer(i4), parameter, public `mo_mrm_constants::nroutingstates` = 2
- integer(i4), parameter, public `mo_mrm_constants::maxnogauges` = 50_i4
- real(dp), parameter, public `mo_mrm_constants::rout_space_weight` = 0._dp
- real(dp), parameter, public `mo_mrm_constants::deltah` = 5.000_dp
- real(dp), dimension(19), parameter `mo_mrm_constants::given_ts` = (/ 60._dp, 120._dp, 180._dp, 240._dp, 300._dp, 360._dp, 600._dp, 720._dp, 900._dp, 1200._dp, 1800._dp, 3600._dp, 7200._dp, 10800._dp, 14400._dp, 21600._dp, 28800._dp, 43200._dp, 86400._dp /)

17.57 mo_mrm_eval.f90 File Reference

Modules

- module `mo_mrm_eval`
Runs mrm with a specific parameter set and returns required variables, e.g. runoff.

Functions/Subroutines

- subroutine, public `mo_mrm_eval::mrm_eval` (parameterset, runoff, sm_opti, basin_avg_tws, neutrons_opti, et_opti)
Runs mrm with a specific parameter set and returns required variables, e.g. runoff.

17.58 mo_mrm_file.f90 File Reference

Modules

- module `mo_mrm_file`
Provides file names and units for mRM.

Variables

- character(len=*), parameter `mo_mrm_file::version` = '1.1'
Current mHM model version.
- character(len=*), parameter `mo_mrm_file::version_date` = 'Nov 2016'
Time of current mHM model version release.
- character(len=*), parameter `mo_mrm_file::file_main` = 'mrm_driver.f90'
Driver file.
- character(len=*), parameter `mo_mrm_file::file_namelist_mrm` = 'mrm.nml'
Namelist file name.
- integer, parameter `mo_mrm_file::unamelist_mrm` = 40
Unit for namelist.
- character(len=*), parameter `mo_mrm_file::file_namelist_param_mrm` = 'mrm_parameter.nml'
Parameter namelists file name.
- integer, parameter `mo_mrm_file::unamelist_param_mrm` = 41
Unit for namelist.
- character(len=*), parameter `mo_mrm_file::file_facc` = 'facc.asc'
• integer, parameter `mo_mrm_file::ufacc` = 56
Unit for flow accumulation input data file.
- character(len=*), parameter `mo_mrm_file::file_fdir` = 'fdir.asc'
flow direction input data file
- integer, parameter `mo_mrm_file::ufdir` = 57
Unit for flow direction input data file.
- character(len=*), parameter `mo_mrm_file::file_gaugeloc` = 'idgauges.asc'
gauge location input data file
- integer, parameter `mo_mrm_file::ugaugeloc` = 62
Unit for gauge location input data file.
- integer, parameter `mo_mrm_file::udischarge` = 66
unit for discharge time series
- character(len=*), parameter `mo_mrm_file::file_defoutput` = 'mrm_outputs.nml'
file defining mRM's outputs
- integer, parameter `mo_mrm_file::udefoutput` = 67
Unit for file defining mRM's outputs.
- character(len=*), parameter `mo_mrm_file::file_config` = 'ConfigFile.log'
file defining mHM's outputs
- integer, parameter `mo_mrm_file::uconfig` = 68
Unit for file defining mHM's outputs.
- character(len=*), parameter `mo_mrm_file::file_daily_discharge` = 'daily_discharge.out'
file defining optimazation outputs
- integer, parameter `mo_mrm_file::udaily_discharge` = 74
Unit for file optimazation outputs.
- character(len=*), parameter `mo_mrm_file::ncfile_discharge` = 'discharge.nc'
file defining optimazation outputs
- character(len=*), parameter `mo_mrm_file::file_mrm_output` = 'mRM_Fluxes_States.nc'
file containing mrm output

17.59 mo_mrm_global_variables.f90 File Reference

Data Types

- type `mo_mrm_global_variables::gaugingstation`
- type `mo_mrm_global_variables::basininfo_mrm`

Modules

- module [mo_mrm_global_variables](#)

Global variables for mRM only.

Variables

- logical [mo_mrm_global_variables::is_start](#)
- integer(i4) [mo_mrm_global_variables::timestep_model_outputs_mrm](#)
- logical, dimension(noutflxstate) [mo_mrm_global_variables::outputflxstate_mrm](#)
- character(256), dimension(:, allocatable, public [mo_mrm_global_variables::dirgauges](#)
- character(256), dimension(:, allocatable, public [mo_mrm_global_variables::dirtotalrunoff](#)
- character(256), public [mo_mrm_global_variables::filenametotalrunoff](#)
- character(256), public [mo_mrm_global_variables::varnametotalrunoff](#)
- type(grid), dimension(:, allocatable, target, public [mo_mrm_global_variables::level11](#)
- type(gridremapper), dimension(:, allocatable, public [mo_mrm_global_variables::l0_l11_remap](#)
- type(gridremapper), dimension(:, allocatable, public [mo_mrm_global_variables::l1_l11_remap](#)
- real(dp), dimension(:, :, allocatable, public [mo_mrm_global_variables::mrm_runoff](#)
- integer(i4), public [mo_mrm_global_variables::ngaugestotal](#)
- integer(i4), public [mo_mrm_global_variables::ninfowgaugestotal](#)
- integer(i4), public [mo_mrm_global_variables::nmeasperday](#)
- type(gaugingstation), public [mo_mrm_global_variables::gauge](#)
- type(gaugingstation), public [mo_mrm_global_variables::inflowgauge](#)
- type(basininfo_mrm), dimension(:, allocatable, target, public [mo_mrm_global_variables::basin_mrm](#)
- integer(i4), dimension(:, allocatable, public [mo_mrm_global_variables::l0_gaugeloc](#)
- integer(i4), dimension(:, allocatable, public [mo_mrm_global_variables::l0_inflowgaugeloc](#)
- integer(i4), dimension(:, allocatable, public [mo_mrm_global_variables::l0_facc](#)
- integer(i4), dimension(:, allocatable, public [mo_mrm_global_variables::l0_fdir](#)
- integer(i4), dimension(:, allocatable, public [mo_mrm_global_variables::l0_drasc](#)
- integer(i4), dimension(:, allocatable, public [mo_mrm_global_variables::l0_dracell](#)
- integer(i4), dimension(:, allocatable, public [mo_mrm_global_variables::l0_streamnet](#)
- integer(i4), dimension(:, allocatable, public [mo_mrm_global_variables::l0_floodplain](#)
- integer(i4), dimension(:, allocatable, public [mo_mrm_global_variables::l11_l1_id](#)
- real(dp), dimension(:, :, allocatable, public [mo_mrm_global_variables::l1_total_runoff_in](#)
- integer(i4), dimension(:, allocatable, public [mo_mrm_global_variables::l1_l11_id](#)
- integer(i4), dimension(:, allocatable, public [mo_mrm_global_variables::l11_fdir](#)
- integer(i4), dimension(:, allocatable, public [mo_mrm_global_variables::l11_noutlets](#)
- integer(i4), dimension(:, allocatable, public [mo_mrm_global_variables::l11_rowout](#)
- integer(i4), dimension(:, allocatable, public [mo_mrm_global_variables::l11_colout](#)
- real(dp), dimension(:, allocatable, public [mo_mrm_global_variables::l11_qmod](#)
- real(dp), dimension(:, allocatable, public [mo_mrm_global_variables::l11_qout](#)
- real(dp), dimension(:, :, allocatable, public [mo_mrm_global_variables::l11_qtin](#)
- real(dp), dimension(:, :, allocatable, public [mo_mrm_global_variables::l11_qtr](#)
- integer(i4), dimension(:, allocatable, public [mo_mrm_global_variables::l11_fromn](#)
- integer(i4), dimension(:, allocatable, public [mo_mrm_global_variables::l11_ton](#)
- integer(i4), dimension(:, allocatable, public [mo_mrm_global_variables::l11_netperm](#)
- integer(i4), dimension(:, allocatable, public [mo_mrm_global_variables::l11_frow](#)
- integer(i4), dimension(:, allocatable, public [mo_mrm_global_variables::l11_fcol](#)
- integer(i4), dimension(:, allocatable, public [mo_mrm_global_variables::l11_trow](#)
- integer(i4), dimension(:, allocatable, public [mo_mrm_global_variables::l11_tcol](#)
- integer(i4), dimension(:, allocatable, public [mo_mrm_global_variables::l11_order](#)
- integer(i4), dimension(:, allocatable, public [mo_mrm_global_variables::l11_label](#)
- logical, dimension(:, allocatable, public [mo_mrm_global_variables::l11_sink](#)
- real(dp), dimension(:, allocatable, public [mo_mrm_global_variables::l11_length](#)

- `real(dp), dimension(:), allocatable, target, public mo_mrm_global_variables::l11_afloodplain`
- `real(dp), dimension(:), allocatable, public mo_mrm_global_variables::l11_slope`
- `real(dp), dimension(:, :), allocatable, public mo_mrm_global_variables::l11_nlinkfracfpimp`
- `real(dp), dimension(:), allocatable, public mo_mrm_global_variables::l11_k`
- `real(dp), dimension(:), allocatable, public mo_mrm_global_variables::l11_xi`
- `real(dp), dimension(:), allocatable, public mo_mrm_global_variables::l11_tsrou`
- `real(dp), dimension(:), allocatable, public mo_mrm_global_variables::l11_c1`
- `real(dp), dimension(:), allocatable, public mo_mrm_global_variables::l11_c2`

17.60 mo_mrm_init.f90 File Reference

Modules

- module `mo_mrm_init`
Wrapper for initializing Routing.

Functions/Subroutines

- subroutine, public `mo_mrm_init::mrm_init` (`file_namelist, unamelist, file_namelist_param, unamelist_param`)
Initialize all mRM variables at all levels (i.e., L0, L1, and L11).
- subroutine `mo_mrm_init::print_startup_message` (`file_namelist, file_namelist_param`)
TODO: add description.
- subroutine `mo_mrm_init::config_output`
TODO: add description.
- subroutine, public `mo_mrm_init::variables_default_init_routing`
Default initialization mRM related L11 variables.
- subroutine `mo_mrm_init::l0_check_input_routing` (`L0Basin_iBasin`)
TODO: add description.
- subroutine `mo_mrm_init::variables_alloc_routing` (`iBasin`)
TODO: add description.
- subroutine `mo_mrm_init::mrm_init_param` (`iBasin, param`)
TODO: add description.
- subroutine, public `mo_mrm_init::mrm_update_param` (`iBasin, param`)
TODO: add description.

17.61 mo_mrm_mpr.f90 File Reference

Modules

- module `mo_mrm_mpr`
Perform Multiscale Parameter Regionalization on Routing Parameters.

Functions/Subroutines

- subroutine, public `mo_mrm_mpr::reg_rout` (`param, length, slope, fFPimp, TS, C1, C2`)
Regionalized routing.

17.62 mo_mrm_net_startup.f90 File Reference

Modules

- module [mo_mrm_net_startup](#)
Startup drainage network for mHM.

Functions/Subroutines

- subroutine, public [mo_mrm_net_startup::l11_l1_mapping](#) (iBasin)
TODO: add description.
- subroutine, public [mo_mrm_net_startup::l11_flow_direction](#) (iBasin)
Determine the flow direction of the upscaled river network at level L11.
- subroutine, public [mo_mrm_net_startup::l11_set_network_topology](#) (iBasin)
Set network topology.
- subroutine, public [mo_mrm_net_startup::l11_routing_order](#) (iBasin)
Find routing order, headwater cells and sink.
- subroutine, public [mo_mrm_net_startup::l11_link_location](#) (iBasin)
Estimate the LO (row,col) location for each routing link at level L11.
- subroutine, public [mo_mrm_net_startup::l11_set_drain_outlet_gauges](#) (iBasin)
Draining cell identification and Set gauging node.
- subroutine, public [mo_mrm_net_startup::l11_stream_features](#) (iBasin)
Stream features (stream network and floodplain)
- subroutine, public [mo_mrm_net_startup::l11_fraction_sealed_floodplain](#) (LCClassImp, do_init)
Fraction of the flood plain with impervious cover.
- subroutine [mo_mrm_net_startup::moveup](#) (elev0, fDir0, fi, fj, ss, nn)
TODO: add description.
- subroutine [mo_mrm_net_startup::movedownonecell](#) (fDir, iRow, jCol)
TODO: add description.
- subroutine [mo_mrm_net_startup::celllength](#) (iBasin, fDir, iRow, jCol, iCoorSystem, length)
TODO: add description.
- subroutine, public [mo_mrm_net_startup::get_distance_two_lat_lon_points](#) (lat1, long1, lat2, long2, distance_out)
estimate distance in [m] between two points in a lat-lon

17.63 mo_mrm_objective_function_runoff.f90 File Reference

Modules

- module [mo_mrm_objective_function_runoff](#)
Objective Functions for Optimization of mHM/mRM against runoff.

Functions/Subroutines

- real(dp) function, public [mo_mrm_objective_function_runoff::single_objective_runoff](#) (parameterset, eval, arg1, arg2, arg3)
Wrapper for objective functions optimizing against runoff.
- subroutine, public [mo_mrm_objective_function_runoff::multi_objective_runoff](#) (parameterset, eval, multi← objectives)
Wrapper for multi-objective functions where at least one is regarding runoff.

- `real(dp) function mo_mrm_objective_function_runoff::loglikelihood_stddev` (parameterset, eval, `stddev`, `stddev_new`, `likeli_new`)

Logarithmic likelihood function with removed linear trend and Lag(1)-autocorrelation.
- `real(dp) function mo_mrm_objective_function_runoff::loglikelihood_evin2013_2` (parameterset, eval, regularize)

Logarithmised likelihood with linear error model and lag(1)-autocorrelation of the relative errors.
- `real(dp) function mo_mrm_objective_function_runoff::parameter_regularization` (paraset, prior, bounds, mask)

TODO: add description.
- `real(dp) function mo_mrm_objective_function_runoff::loglikelihood_trend_no_autocorr` (parameterset, eval, `stddev_old`, `stddev_new`, `likeli_new`)

Logarithmic likelihood function with linear trend removed.
- `real(dp) function mo_mrm_objective_function_runoff::objective_lnnse` (parameterset, eval)

Objective function of logarithmic NSE.
- `real(dp) function mo_mrm_objective_function_runoff::objective_sse` (parameterset, eval)

Objective function of SSE.
- `real(dp) function mo_mrm_objective_function_runoff::objective_nse` (parameterset, eval)

Objective function of NSE.
- `real(dp) function mo_mrm_objective_function_runoff::objective_equal_nse_lnnse` (parameterset, eval)

Objective function equally weighting NSE and lnNSE.
- `real(dp) function, dimension(2) mo_mrm_objective_function_runoff::multi_objective_nse_lnnse` (parameterset, eval)

Multi-objective function with NSE and lnNSE.
- `real(dp) function, dimension(2) mo_mrm_objective_function_runoff::multi_objective_lnnse_highflow_lnnse_lowflow` (parameterset, eval)

Multi-objective function with NSE and lnNSE.
- `real(dp) function, dimension(2) mo_mrm_objective_function_runoff::multi_objective_lnnse_highflow_lnnse_lowflow_2` (parameterset, eval)

Multi-objective function with NSE and lnNSE.
- `real(dp) function, dimension(2) mo_mrm_objective_function_runoff::multi_objective_ae_fdc_lsv_nse_djf` (parameterset, eval)

Multi-objective function with absolute error of Flow Duration Curves low-segment volume and nse of DJF's discharge.
- `real(dp) function mo_mrm_objective_function_runoff::objective_power6_nse_lnnse` (parameterset, eval)

Objective function of combined NSE and lnNSE with power of 5 i.e. the p-norm with p=5.
- `real(dp) function mo_mrm_objective_function_runoff::objective_kge` (parameterset, eval)

Objective function of KGE.
- `real(dp) function mo_mrm_objective_function_runoff::objective_multiple_gauges_kge_power6` (parameterset, eval)

combined objective function based on KGE raised to the power 6
- `real(dp) function mo_mrm_objective_function_runoff::objective_weighted_nse` (parameterset, eval)

Objective function of weighted NSE.
- subroutine, public `mo_mrm_objective_function_runoff::extract_runoff` (`gaugeld`, `runoff`, `runoff_agg`, `runoff_←obs`, `runoff_obs_mask`)

extracts runoff data from global variables

17.64 mo_mrm_read_config.f90 File Reference

Modules

- module `mo_mrm_read_config`

read mRM config

Functions/Subroutines

- subroutine, public `mo_mrm_read_config::mrm_read_config` (`file_namelist`, `unamelist`, `file_namelist_param`, `unamelist_param`, `do_message`, `readLatLon`)
Read the general config of mRM.
- subroutine `mo_mrm_read_config::read_mrm_routing_params` (`processCase`, `file_namelist_param`, `unamelist_param`)
TODO: add description.

17.65 mo_mrm_read_data.f90 File Reference

Modules

- module `mo_mrm_read_data`
This module contains all routines to read mRM data from file.

Functions/Subroutines

- subroutine, public `mo_mrm_read_data::mrm_read_l0_data` (`do_reinit`, `do_readlatlon`, `do_readcover`)
read L0 data from file
- subroutine, public `mo_mrm_read_data::mrm_read_discharge`
Read discharge timeseries from file.
- subroutine, public `mo_mrm_read_data::mrm_read_total_runoff` (`iBasin`)
read simulated runoff that is to be routed
- subroutine `mo_mrm_read_data::rotate_fdir_variable` (`x`)
TODO: add description.

17.66 mo_mrm_restart.f90 File Reference

Modules

- module `mo_mrm_restart`
Restart routines.

Functions/Subroutines

- subroutine, public `mo_mrm_restart::mrm_write_restart` (`iBasin`, `OutPath`)
write routing states and configuration
- subroutine, public `mo_mrm_restart::mrm_read_restart_states` (`iBasin`, `InPath`)
read routing states
- subroutine, public `mo_mrm_restart::mrm_read_restart_config` (`iBasin`, `InPath`)
reads Level 11 configuration from a restart directory

17.67 mo_mrm_routing.f90 File Reference

Modules

- module `mo_mrm_routing`
Performs runoff routing for mHM at level L11.

Functions/Subroutines

- subroutine, public `mo_mrm_routing::mrm_routing` (read_states, processCase, global_routing_param, L1 \leftarrow _total_runoff, L1_areaCell, L1_L11_Id, L11_areaCell, L11_L1_Id, L11_netPerm, L11_fromN, L11_toN, L11_nOutlets, timestep, tsRoutFactor, nNodes, nInflowGauges, InflowGaugeIndexList, InflowGauge \leftarrow Headwater, InflowGaugeNodeList, InflowDischarge, nGauges, gaugeIndexList, gaugeNodeList, map_flag, L11_length, L11_slope, L11_FracFPimp, L11_C1, L11_C2, L11_qOut, L11_qTIN, L11_qTR, L11_qMod, GaugeDischarge)

route water given runoff
- subroutine `mo_mrm_routing::l11_runoff_acc` (qAll, efecArea, L1_L11_Id, L11_areaCell, L11_L1_Id, TS, map_flag, qAcc)

total runoff accumulation at L11.
- subroutine `mo_mrm_routing::add_inflow` (nInflowGauges, InflowIndexList, InflowHeadwater, InflowNodeList, QInflow, qOut)

Adds inflow discharge to the runoff produced at the cell where the inflow is occurring.
- subroutine `mo_mrm_routing::l11_routing` (nNodes, nLinks, netPerm, netLink_fromN, netLink_toN, netLink \leftarrow _C1, netLink_C2, netNode_qOUT, nInflowGauges, InflowHeadwater, InflowNodeList, netNode_qTIN, net \leftarrow Node_qTR, netNode_Qmod)

Performs runoff routing for mHM at L11 upscaled network ([Routing Network](#)).

17.68 `mo_mrm_signatures.f90` File Reference

Modules

- module `mo_mrm_signatures`

Module with calculations for several hydrological signatures.

Functions/Subroutines

- real(dp) function, dimension(size(lags, 1)), public `mo_mrm_signatures::autocorrelation` (data, lags, mask)

Autocorrelation of a given data series.
- real(dp) function, dimension(size(quantiles, 1)), public `mo_mrm_signatures::flowdurationcurve` (data, quantiles, mask, concavity_index, mid_segment_slope, mhigh_segment_volume, high_segment_volume, low \leftarrow segment_volume)

Flow duration curves.
- subroutine, public `mo_mrm_signatures::limb_densities` (data, mask, RLD, DLD)

Calculates limb densities.
- real(dp) function `mo_mrm_signatures::maximummonthlyflow` (data, mask, yr_start, mo_start, dy_start)

Maximum of average flows per months.
- subroutine, public `mo_mrm_signatures::moments` (data, mask, mean_data, stddev_data, median_data, max_data, mean_log, stddev_log, median_log, max_log)

Moments of data and log-transformed data, e.g. mean and standard deviation.
- real(dp) function, dimension(size(quantiles, 1)), public `mo_mrm_signatures::peakdistribution` (data, quantiles, mask, slope_peak_distribution)

Calculates the peak distribution.
- real(dp) function, public `mo_mrm_signatures::runoffratio` (data, basin_area, mask, precip_series, precip \leftarrow sum, log_data)

Runoff ratio (accumulated daily discharge [mm/d] / accumulated daily precipitation [mm/d]).
- real(dp) function, public `mo_mrm_signatures::zeroflowratio` (data, mask)

Ratio of zero values to total number of data points.

17.69 mo_mrm_write.f90 File Reference

Modules

- module [mo_mrm_write](#)
write of discharge and restart files

Functions/Subroutines

- subroutine, public [mo_mrm_write::mrm_write](#)
write discharge and restart files
- subroutine [mo_mrm_write::write_configfile](#)
This module writes the results of the configuration into an ASCII-file.
- subroutine [mo_mrm_write::write_daily_obs_sim_discharge](#) (Qobs, Qsim)
Write a file for the daily observed and simulated discharge timeseries during the evaluation period for each gauging station.
- subroutine, public [mo_mrm_write::mrm_write_output_fluxes](#) (iBasin, nCells, timeStep_model_outputs, warmingDays, newTime, nTimeSteps, nTStepDay, tt, day, month, year, timestep, mask11, L11_qmod)
write fluxes to netcdf output files
- subroutine, public [mo_mrm_write::mrm_write_optifile](#) (best_OF, best_paramSet, param_names)
Write briefly final optimization results.
- subroutine, public [mo_mrm_write::mrm_write_optinamelist](#) (parameters, maskpara, parameters_name)
Write final, optimized parameter set in a namelist format.

Variables

- integer(i4) [mo_mrm_write::day_counter](#)
- integer(i4) [mo_mrm_write::month_counter](#)
- integer(i4) [mo_mrm_write::year_counter](#)
- integer(i4) [mo_mrm_write::average_counter](#)
- type(outputdataset) [mo_mrm_write::nc](#)

17.70 mo_mrm_write_fluxes_states.f90 File Reference

Data Types

- interface [mo_mrm_write_fluxes_states::outputvariable](#)
- interface [mo_mrm_write_fluxes_states::outputvariable](#)
- interface [mo_mrm_write_fluxes_states::outputdataset](#)
- interface [mo_mrm_write_fluxes_states::outputdataset](#)

Modules

- module [mo_mrm_write_fluxes_states](#)
Creates NetCDF output for different fluxes and state variables of mHM.

Functions/Subroutines

- type(outputvariable) function `mo_mrm_write_fluxes_states::newoutputvariable` (nc, name, dtype, dims, ncells, mask, avg)

Initialize OutputVariable.
- subroutine `mo_mrm_write_fluxes_states::updatevariable` (self, data)

Update OutputVariable.
- subroutine `mo_mrm_write_fluxes_states::writevariabletimestep` (self, timestep)

Write timestep to file.
- type(outputdataset) function `mo_mrm_write_fluxes_states::newoutputdataset` (ibasin, mask, nCells)

Initialize OutputDataset.
- subroutine `mo_mrm_write_fluxes_states::updatedataset` (self, sidx, eidx, L11_Qmod)

Update all variables.
- subroutine `mo_mrm_write_fluxes_states::writetimestep` (self, timestep)

Write all accumulated data.
- subroutine `mo_mrm_write_fluxes_states::close` (self)

Close the file.
- type(ncdataset) function `mo_mrm_write_fluxes_states::createoutputfile` (ibasin)

Create and initialize output file.
- subroutine `mo_mrm_write_fluxes_states::writevariableattributes` (var, long_name, unit)

Write output variable attributes.

17.71 mo_multi_param_reg.f90 File Reference

Modules

- module `mo_multi_param_reg`

Multiscale parameter regionalization (MPR).

Functions/Subroutines

- subroutine, public `mo_multi_param_reg::mpr` (mask0, geoUnit0, soilld0, Asp0, gridded_LAI0, LCover0, slope_emp0, y0, Id0, upper_bound1, lower_bound1, left_bound1, right_bound1, n_subcells1, fSealed1, alpha1, degDayInc1, degDayMax1, degDayNoPre1, fAsp1, HarSamCoeff1, PrieTayAlpha1, aeroResist1, surfResist1, fRoots1, kFastFlow1, kSlowFlow1, kBaseFlow1, kPerco1, karstLoss1, soilMoistFC1, soilMoistSat1, soilMoistExp1, jarvis_thresh_c1, tempThresh1, unsatThresh1, sealedThresh1, wiltingPoint1, maxInter1, petLAIcorFactor, parameterset)

Regionalizing and Upscaling process parameters.
- subroutine `mo_multi_param_reg::baseflow_param` (param, geoUnit0, k2_0)

baseflow recession parameter
- subroutine `mo_multi_param_reg::snow_acc_melt_param` (param, c2TSTu, fForest1, flperm1, fPerm1, tempThresh1, degDayNoPre1, degDayInc1, degDayMax1)

Calculates the snow parameters.
- subroutine `mo_multi_param_reg::iper_thres_runoff` (param, sealedThresh1)

sets the impervious layer threshold parameter for runoff generation
- subroutine `mo_multi_param_reg::karstic_layer` (param, geoUnit0, mask0, SMs_FC0, KsVar_V0, Id0, n_subcells1, upper_bound1, lower_bound1, left_bound1, right_bound1, karstLoss1, L1_Kp)

calculates the Karstic percolation loss
- subroutine, public `mo_multi_param_reg::canopy_intercept_param` (processMatrix, param, LAI0, n_subcells1, upper_bound1, lower_bound1, left_bound1, right_bound1, Id0, mask0, nodata, max_intercept1)

estimate effective maximum interception capacity at L1

- subroutine [mo_multi_param_reg::aerodynamical_resistance](#) (LAI0, LCover0, param, mask0, Id0, n←subcells1, upper_bound1, lower_bound1, left_bound1, right_bound1, aerodyn_resistance1)

Regionalization of aerodynamic resistance.

17.72 mo_ncread.f90 File Reference

Data Types

- interface [mo_ncread::get_ncvar](#)

Modules

- module [mo_ncread](#)

Functions/Subroutines

- integer(i4) function, dimension(5), public [mo_ncread::get_ncdim](#) (Filename, Variable, PrintInfo, ndims)
- subroutine, public [mo_ncread::get_ncdimatt](#) (Filename, Variable, DimName, DimLen)
- subroutine, public [mo_ncread::get_ncvaratt](#) (FileName, VarName, AttName, AttValues, fid, dtype)
- subroutine [mo_ncread::get_ncvar_0d_sp](#) (Filename, VarName, Dat, fid)
- subroutine [mo_ncread::get_ncvar_0d_dp](#) (Filename, VarName, Dat, fid)
- subroutine [mo_ncread::get_ncvar_1d_sp](#) (Filename, VarName, Dat, start, a_count, fid)
- subroutine [mo_ncread::get_ncvar_1d_dp](#) (Filename, VarName, Dat, start, a_count, fid)
- subroutine [mo_ncread::get_ncvar_2d_sp](#) (Filename, VarName, Dat, start, a_count, fid)
- subroutine [mo_ncread::get_ncvar_2d_dp](#) (Filename, VarName, Dat, start, a_count, fid)
- subroutine [mo_ncread::get_ncvar_3d_sp](#) (Filename, VarName, Dat, start, a_count, fid)
- subroutine [mo_ncread::get_ncvar_3d_dp](#) (Filename, VarName, Dat, start, a_count, fid)
- subroutine [mo_ncread::get_ncvar_4d_sp](#) (Filename, VarName, Dat, start, a_count, fid)
- subroutine [mo_ncread::get_ncvar_4d_dp](#) (Filename, VarName, Dat, start, a_count, fid)
- subroutine [mo_ncread::get_ncvar_5d_sp](#) (Filename, VarName, Dat, start, a_count, fid)
- subroutine [mo_ncread::get_ncvar_5d_dp](#) (Filename, VarName, Dat, start, a_count, fid)
- subroutine [mo_ncread::get_ncvar_0d_i4](#) (Filename, VarName, Dat, fid)
- subroutine [mo_ncread::get_ncvar_1d_i4](#) (Filename, VarName, Dat, start, a_count, fid)
- subroutine [mo_ncread::get_ncvar_2d_i4](#) (Filename, VarName, Dat, start, a_count, fid)
- subroutine [mo_ncread::get_ncvar_3d_i4](#) (Filename, VarName, Dat, start, a_count, fid)
- subroutine [mo_ncread::get_ncvar_4d_i4](#) (Filename, VarName, Dat, start, a_count, fid)
- subroutine [mo_ncread::get_ncvar_5d_i4](#) (Filename, VarName, Dat, start, a_count, fid)
- subroutine [mo_ncread::get_ncvar_0d_i1](#) (Filename, VarName, Dat, fid)
- subroutine [mo_ncread::get_ncvar_1d_i1](#) (Filename, VarName, Dat, start, a_count, fid)
- subroutine [mo_ncread::get_ncvar_2d_i1](#) (Filename, VarName, Dat, start, a_count, fid)
- subroutine [mo_ncread::get_ncvar_3d_i1](#) (Filename, VarName, Dat, start, a_count, fid)
- subroutine [mo_ncread::get_ncvar_4d_i1](#) (Filename, VarName, Dat, start, a_count, fid)
- subroutine [mo_ncread::get_ncvar_5d_i1](#) (Filename, VarName, Dat, start, a_count, fid)
- integer(i4) function, public [mo_ncread::ncopen](#) (Fname)
- subroutine, public [mo_ncread::ncclose](#) (ncid)
- subroutine [mo_ncread::get_info](#) (Varname, ncid, varid, xtype, dl, Info, ndims)
- subroutine [mo_ncread::check](#) (status)

17.73 mo_ncwrite.f90 File Reference

Data Types

- type `mo_ncwrite::dims`
- type `mo_ncwrite::attribute`
- type `mo_ncwrite::variable`
- interface `mo_ncwrite::dump_netcdf`
- interface `mo_ncwrite::var2nc`

Extended `dump_netcdf` for multiple variables.

Modules

- module `mo_ncwrite`

Functions/Subroutines

- subroutine, public `mo_ncwrite::close_netcdf` (`ncid`)
- subroutine, public `mo_ncwrite::create_netcdf` (`filename`, `ncid`, `lfs`, `netcdf4`, `deflate_level`)
- subroutine `mo_ncwrite::dump_netcdf_1d_sp` (`filename`, `arr`, `append`, `lfs`, `netcdf4`, `deflate_level`)
- subroutine `mo_ncwrite::dump_netcdf_2d_sp` (`filename`, `arr`, `append`, `lfs`, `netcdf4`, `deflate_level`)
- subroutine `mo_ncwrite::dump_netcdf_3d_sp` (`filename`, `arr`, `append`, `lfs`, `netcdf4`, `deflate_level`)
- subroutine `mo_ncwrite::dump_netcdf_4d_sp` (`filename`, `arr`, `append`, `lfs`, `netcdf4`, `deflate_level`)
- subroutine `mo_ncwrite::dump_netcdf_5d_sp` (`filename`, `arr`, `append`, `lfs`, `netcdf4`, `deflate_level`)
- subroutine `mo_ncwrite::dump_netcdf_1d_dp` (`filename`, `arr`, `append`, `lfs`, `netcdf4`, `deflate_level`)
- subroutine `mo_ncwrite::dump_netcdf_2d_dp` (`filename`, `arr`, `append`, `lfs`, `netcdf4`, `deflate_level`)
- subroutine `mo_ncwrite::dump_netcdf_3d_dp` (`filename`, `arr`, `append`, `lfs`, `netcdf4`, `deflate_level`)
- subroutine `mo_ncwrite::dump_netcdf_4d_dp` (`filename`, `arr`, `append`, `lfs`, `netcdf4`, `deflate_level`)
- subroutine `mo_ncwrite::dump_netcdf_5d_dp` (`filename`, `arr`, `append`, `lfs`, `netcdf4`, `deflate_level`)
- subroutine `mo_ncwrite::dump_netcdf_1d_i4` (`filename`, `arr`, `append`, `lfs`, `netcdf4`, `deflate_level`)
- subroutine `mo_ncwrite::dump_netcdf_2d_i4` (`filename`, `arr`, `append`, `lfs`, `netcdf4`, `deflate_level`)
- subroutine `mo_ncwrite::dump_netcdf_3d_i4` (`filename`, `arr`, `append`, `lfs`, `netcdf4`, `deflate_level`)
- subroutine `mo_ncwrite::dump_netcdf_4d_i4` (`filename`, `arr`, `append`, `lfs`, `netcdf4`, `deflate_level`)
- subroutine `mo_ncwrite::dump_netcdf_5d_i4` (`filename`, `arr`, `append`, `lfs`, `netcdf4`, `deflate_level`)
- subroutine `mo_ncwrite::var2nc_1d_i4` (`f_name`, `arr`, `dnames`, `v_name`, `dim_unlimited`, `long_name`, `units`, `missing_value`, `attributes`, `create`, `ncid`, `nrec`)
- subroutine `mo_ncwrite::var2nc_1d_sp` (`f_name`, `arr`, `dnames`, `v_name`, `dim_unlimited`, `long_name`, `units`, `missing_value`, `attributes`, `create`, `ncid`, `nrec`)
- subroutine `mo_ncwrite::var2nc_1d_dp` (`f_name`, `arr`, `dnames`, `v_name`, `dim_unlimited`, `long_name`, `units`, `missing_value`, `attributes`, `create`, `ncid`, `nrec`)
- subroutine `mo_ncwrite::var2nc_2d_i4` (`f_name`, `arr`, `dnames`, `v_name`, `dim_unlimited`, `long_name`, `units`, `missing_value`, `attributes`, `create`, `ncid`, `nrec`)
- subroutine `mo_ncwrite::var2nc_2d_sp` (`f_name`, `arr`, `dnames`, `v_name`, `dim_unlimited`, `long_name`, `units`, `missing_value`, `attributes`, `create`, `ncid`, `nrec`)
- subroutine `mo_ncwrite::var2nc_2d_dp` (`f_name`, `arr`, `dnames`, `v_name`, `dim_unlimited`, `long_name`, `units`, `missing_value`, `attributes`, `create`, `ncid`, `nrec`)
- subroutine `mo_ncwrite::var2nc_3d_i4` (`f_name`, `arr`, `dnames`, `v_name`, `dim_unlimited`, `long_name`, `units`, `missing_value`, `attributes`, `create`, `ncid`, `nrec`)
- subroutine `mo_ncwrite::var2nc_3d_sp` (`f_name`, `arr`, `dnames`, `v_name`, `dim_unlimited`, `long_name`, `units`, `missing_value`, `attributes`, `create`, `ncid`, `nrec`)
- subroutine `mo_ncwrite::var2nc_3d_dp` (`f_name`, `arr`, `dnames`, `v_name`, `dim_unlimited`, `long_name`, `units`, `missing_value`, `attributes`, `create`, `ncid`, `nrec`)
- subroutine `mo_ncwrite::var2nc_4d_i4` (`f_name`, `arr`, `dnames`, `v_name`, `dim_unlimited`, `long_name`, `units`, `missing_value`, `attributes`, `create`, `ncid`, `nrec`)

- subroutine `mo_ncwrite::var2nc_4d_sp` (f_name, arr, dnames, v_name, dim_unlimited, long_name, units, missing_value, attributes, create, ncid, nrec)
- subroutine `mo_ncwrite::var2nc_4d_dp` (f_name, arr, dnames, v_name, dim_unlimited, long_name, units, missing_value, attributes, create, ncid, nrec)
- subroutine `mo_ncwrite::var2nc_5d_i4` (f_name, arr, dnames, v_name, dim_unlimited, long_name, units, missing_value, attributes, create, ncid, nrec)
- subroutine `mo_ncwrite::var2nc_5d_sp` (f_name, arr, dnames, v_name, dim_unlimited, long_name, units, missing_value, attributes, create, ncid, nrec)
- subroutine `mo_ncwrite::var2nc_5d_dp` (f_name, arr, dnames, v_name, dim_unlimited, long_name, units, missing_value, attributes, create, ncid, nrec)
- subroutine, public `mo_ncwrite::write_dynamic_ncdf` (ncl, irec)
- subroutine, public `mo_ncwrite::write_static_ncdf` (ncl)
- integer(i4) function `mo_ncwrite::open_ncdf` (f_name, create)
- subroutine `mo_ncwrite::check` (status)

Variables

- integer(i4), parameter, public `mo_ncwrite::nmaxdim` = 5
- integer(i4), parameter, public `mo_ncwrite::nmaxatt` = 20
- integer(i4), parameter, public `mo_ncwrite::maxlen` = 256
- integer(i4), parameter, public `mo_ncwrite::ngatt` = 20
- integer(i4), parameter, public `mo_ncwrite::nattdim` = 2
- integer(i4), public `mo_ncwrite::nvars`
- integer(i4), public `mo_ncwrite::ndims`
- type(dims), dimension(:), allocatable, public `mo_ncwrite::dnc`
- type(variable), dimension(:), allocatable, public `mo_ncwrite::v`
- type(attribute), dimension(ngatt), public `mo_ncwrite::gatt`

17.74 mo_ncdf.f90 File Reference

Data Types

- interface `mo_ncdf::ncdataset`
Provides basic file modification functionality.
- interface `mo_ncdf::ncdataset`
Provides basic file modification functionality.
- type `mo_ncdf::ncdimension`
Provides the dimension access functionality.
- interface `mo_ncdf::ncvariable`

Modules

- module `mo_ncdf`
NetCDF Fortran 90 interface wrapper.

Functions/Subroutines

- subroutine `mo_ncdf::initncvariable` (self, id, parent)
- subroutine `mo_ncdf::initncdimension` (self, id, parent)
- subroutine `mo_ncdf::initncdataset` (self, fname, mode)
- type(ncvariable) function `mo_ncdf::newncvariable` (id, parent)

- type(ncdimension) function `mo_netcdf::newncdimension` (id, parent)
- type(ncdataset) function `mo_netcdf::newncdataset` (fname, mode)
- subroutine `mo_netcdf::close` (self)
- integer(i4) function `mo_netcdf::getnovariables` (self)
- integer(i4) function, dimension(:,), allocatable `mo_netcdf::getvariableids` (self)
- type(ncvariable) function, dimension(:,), allocatable `mo_netcdf::getvariables` (self)
- character(len=256) function `mo_netcdf::getdimensionname` (self)
- integer(i4) function `mo_netcdf::getdimensionlength` (self)
- logical function `mo_netcdf::isdatasetunlimited` (self)
- type(ncdimension) function `mo_netcdf::getunlimiteddimension` (self)
- logical function `mo_netcdf::equalncdimensions` (dim1, dim2)
- logical function `mo_netcdf::isunlimiteddimension` (self)
- type(ncdimension) function `mo_netcdf::setdimension` (self, name, length)
- logical function `mo_netcdf::hasvariable` (self, name)
- logical function `mo_netcdf::hasdimension` (self, name)
- type(ncvariable) function `mo_netcdf::setvariablewithids` (self, name, dtype, dimensions, contiguous, chunk-sizes, deflate_level, shuffle, fletcher32, endianness, cache_size, cache_nelems, cache_preemption)
- type(ncvariable) function `mo_netcdf::setvariablewithnames` (self, name, dtype, dimensions, contiguous, chunksizes, deflate_level, shuffle, fletcher32, endianness, cache_size, cache_nelems, cache_preemption)
- type(ncvariable) function `mo_netcdf::setvariablewithtypes` (self, name, dtype, dimensions, contiguous, chunk-sizes, deflate_level, shuffle, fletcher32, endianness, cache_size, cache_nelems, cache_preemption)
- type(ncdimension) function `mo_netcdf::getdimensionbyid` (self, id)
- type(ncdimension) function `mo_netcdf::getdimensionbyname` (self, name)
- type(ncvariable) function `mo_netcdf::getvariablebyname` (self, name)
- character(len=256) function `mo_netcdf::getvariablelename` (self)
- integer(i4) function `mo_netcdf::getnondimensions` (self)
- type(ncdimension) function, dimension(:,), allocatable `mo_netcdf::getvariabledimensions` (self)
- integer(i4) function, dimension(:,), allocatable `mo_netcdf::getvariablesshape` (self)
- character(3) function `mo_netcdf::getvariabledtype` (self)
- logical function `mo_netcdf::isunlimitedvariable` (self)
- logical function `mo_netcdf::hasattribute` (self, name)
- subroutine `mo_netcdf::setglobalattributechar` (self, name, data)
- subroutine `mo_netcdf::setglobalattributei8` (self, name, data)
- subroutine `mo_netcdf::setglobalattributei16` (self, name, data)
- subroutine `mo_netcdf::setglobalattributei32` (self, name, data)
- subroutine `mo_netcdf::setglobalattributei64` (self, name, data)
- subroutine `mo_netcdf::setglobalattributef32` (self, name, data)
- subroutine `mo_netcdf::setglobalattributef64` (self, name, data)
- subroutine `mo_netcdf::getglobalattributechar` (self, name, avalue)
- subroutine `mo_netcdf::getglobalattributei8` (self, name, avalue)
- subroutine `mo_netcdf::getglobalattributei16` (self, name, avalue)
- subroutine `mo_netcdf::getglobalattributei32` (self, name, avalue)
- subroutine `mo_netcdf::getglobalattributei64` (self, name, avalue)
- subroutine `mo_netcdf::getglobalattributef32` (self, name, avalue)
- subroutine `mo_netcdf::getglobalattributef64` (self, name, avalue)
- subroutine `mo_netcdf::setvariableattributechar` (self, name, data)
- subroutine `mo_netcdf::setvariableattributei8` (self, name, data)
- subroutine `mo_netcdf::setvariableattributei16` (self, name, data)
- subroutine `mo_netcdf::setvariableattributei32` (self, name, data)
- subroutine `mo_netcdf::setvariableattributei64` (self, name, data)
- subroutine `mo_netcdf::setvariableattributef32` (self, name, data)
- subroutine `mo_netcdf::setvariableattributef64` (self, name, data)
- subroutine `mo_netcdf::getvariableattributechar` (self, name, avalue)
- subroutine `mo_netcdf::getvariableattributei8` (self, name, avalue)
- subroutine `mo_netcdf::getvariableattributei16` (self, name, avalue)

- subroutine `mo_ncdf::getdata3di8` (self, data, start, cnt, stride, map)
- subroutine `mo_ncdf::getdata4di8` (self, data, start, cnt, stride, map)
- subroutine `mo_ncdf::getdata5di8` (self, data, start, cnt, stride, map)
- subroutine `mo_ncdf::getdatascalari16` (self, data, start, cnt, stride, map)
- subroutine `mo_ncdf::getdata1di16` (self, data, start, cnt, stride, map)
- subroutine `mo_ncdf::getdata2di16` (self, data, start, cnt, stride, map)
- subroutine `mo_ncdf::getdata3di16` (self, data, start, cnt, stride, map)
- subroutine `mo_ncdf::getdata4di16` (self, data, start, cnt, stride, map)
- subroutine `mo_ncdf::getdata5di16` (self, data, start, cnt, stride, map)
- subroutine `mo_ncdf::getdatascalari32` (self, data, start, cnt, stride, map)
- subroutine `mo_ncdf::getdata1di32` (self, data, start, cnt, stride, map)
- subroutine `mo_ncdf::getdata2di32` (self, data, start, cnt, stride, map)
- subroutine `mo_ncdf::getdata3di32` (self, data, start, cnt, stride, map)
- subroutine `mo_ncdf::getdata4di32` (self, data, start, cnt, stride, map)
- subroutine `mo_ncdf::getdata5di32` (self, data, start, cnt, stride, map)
- subroutine `mo_ncdf::getdatascalari64` (self, data, start, cnt, stride, map)
- subroutine `mo_ncdf::getdata1di64` (self, data, start, cnt, stride, map)
- subroutine `mo_ncdf::getdata2di64` (self, data, start, cnt, stride, map)
- subroutine `mo_ncdf::getdata3di64` (self, data, start, cnt, stride, map)
- subroutine `mo_ncdf::getdata4di64` (self, data, start, cnt, stride, map)
- subroutine `mo_ncdf::getdata5di64` (self, data, start, cnt, stride, map)
- subroutine `mo_ncdf::getdatascalarf32` (self, data, start, cnt, stride, map)
- subroutine `mo_ncdf::getdata1df32` (self, data, start, cnt, stride, map)
- subroutine `mo_ncdf::getdata2df32` (self, data, start, cnt, stride, map)
- subroutine `mo_ncdf::getdata3df32` (self, data, start, cnt, stride, map)
- subroutine `mo_ncdf::getdata4df32` (self, data, start, cnt, stride, map)
- subroutine `mo_ncdf::getdata5df32` (self, data, start, cnt, stride, map)
- subroutine `mo_ncdf::getdatascalarf64` (self, data, start, cnt, stride, map)
- subroutine `mo_ncdf::getdata1df64` (self, data, start, cnt, stride, map)
- subroutine `mo_ncdf::getdata2df64` (self, data, start, cnt, stride, map)
- subroutine `mo_ncdf::getdata3df64` (self, data, start, cnt, stride, map)
- subroutine `mo_ncdf::getdata4df64` (self, data, start, cnt, stride, map)
- subroutine `mo_ncdf::getdata5df64` (self, data, start, cnt, stride, map)
- integer(i4) function, dimension(datarank) `mo_ncdf::getreaddatashape` (var, datarank, instart, incnt, instride)
- integer(i4) function `mo_ncdf::getdtypefromstring` (dtype)
- character(3) function `mo_ncdf::getdtypefrominteger` (dtype)
- subroutine `mo_ncdf::check` (status, msg)

17.75 mo_neutrons.f90 File Reference

Modules

- module `mo_neutrons`

Models to predict neutron intensities above soils.

Functions/Subroutines

- subroutine, public `mo_neutrons::desiletsn0` (SoilMoisture, Horizons, N0, neutrons)

Calculate neutrons from soil moisture in the first layer.
- subroutine, public `mo_neutrons::cosmic` (SoilMoisture, Horizons, params, neutron_integral_AFast, neutrons)

Calculate neutrons from soil moisture in all layers.
- subroutine `mo_neutrons::oldintegration` (res, c)

- subroutine, public `mo_neutrons::tabularintegralfast` (integral, maxC)

Save approximation data for A_fast.
- subroutine `mo_neutrons::approx_mon_int` (res, f, c, xmin, xmax, eps, steps, fxmin, fxmax)

TODO: add description.
- recursive subroutine `mo_neutrons::approx_mon_int_steps` (res, f, c, xmin, xmax, eps, steps, fxmin, fxmax)

TODO: add description.
- recursive subroutine `mo_neutrons::approx_mon_int_eps` (res, f, c, xmin, xmax, eps, steps, fxmin, fxmax)

TODO: add description.
- subroutine `mo_neutrons::lookupintegral` (res, integral, c)

TODO: add description.
- real(dp) function `mo_neutrons::intgrandfast` (c, phi)

TODO: add description.

17.76 mo_nml.f90 File Reference

Modules

- module `mo_nml`

Deal with namelist files.

Functions/Subroutines

- subroutine, public `mo_nml::open_nml` (file, unit, quiet)

Open a namelist file.
- subroutine, public `mo_nml::close_nml` (unit)

Close a namelist file.
- subroutine, public `mo_nml::position_nml` (name, unit, status, first)

Position a namelist file.

Variables

- integer(i4), parameter, public `mo_nml::positioned` = 0

Information: file pointer set to namelist group.
- integer(i4), parameter, public `mo_nml::missing` = 1

Error: namelist group is missing.
- integer(i4), parameter, public `mo_nml::length_error` = 2

Error: namelist group name too long.
- integer(i4), parameter, public `mo_nml::read_error` = 3

Error occurred during read of namelist file.
- integer, save, public `mo_nml::nunitnml` = -1

17.77 mo_objective_function.f90 File Reference

Modules

- module `mo_objective_function`

Objective Functions for Optimization of mHM.

Functions/Subroutines

- `real(dp) function, public mo_objective_function::objective` (`parameterset, eval, arg1, arg2, arg3`)

Wrapper for objective functions.
- `real(dp) function mo_objective_function::objective_sm_kge_catchment_avg` (`parameterset, eval`)

Objective function for soil moisture.
- `real(dp) function mo_objective_function::objective_sm_corr` (`parameterset, eval`)

Objective function for soil moisture.
- `real(dp) function mo_objective_function::objective_sm_pd` (`parameterset, eval`)

Objective function for soil moisture.
- `real(dp) function mo_objective_function::objective_sm_sse_standard_score` (`parameterset, eval`)

Objective function for soil moisture.
- `real(dp) function mo_objective_function::objective_kge_q_rmse_tws` (`parameterset, eval`)

Objective function of KGE for runoff and RMSE for basin_avg TWS (standarized scores)
- `real(dp) function mo_objective_function::objective_neutrons_kge_catchment_avg` (`parameterset, eval`)

Objective function for neutrons.
- `real(dp) function mo_objective_function::objective_et_kge_catchment_avg` (`parameterset, eval`)

Objective function for evapotranspiration (et).
- `real(dp) function mo_objective_function::objective_kge_q_sm_corr` (`parameterset, eval`)

Objective function of KGE for runoff and correlation for SM.
- `real(dp) function mo_objective_function::objective_kge_q_et` (`parameterset, eval`)

Objective function of KGE for runoff and KGE for ET.
- `real(dp) function mo_objective_function::objective_kge_q_rmse_et` (`parameterset, eval`)

Objective function of KGE for runoff and RMSE for basin_avg ET (standarized scores)
- `subroutine mo_objective_function::extract_basin_avg_tws` (`basinId, tws, tws_sim, tws_obs, tws_obs_mask`)

extracts basin average tws data from global variables

17.78 mo_optimization.f90 File Reference

Modules

- `module mo_optimization`

Wrapper subroutine for optimization against runoff and sm.

Functions/Subroutines

- `subroutine, public mo_optimization::optimization` (`eval, objective, dirConfigOut, funcBest, maskpara`)

Wrapper for optimization.

17.79 mo_optimization_utils.f90 File Reference

Data Types

- `interface mo_optimization_utils::eval_interface`
- `interface mo_optimization_utils::objective_interface`

Modules

- `module mo_optimization_utils`

17.80 mo_orderpack.f90 File Reference

Data Types

- interface `mo_orderpack::sort`
Unconditional ranking.
- interface `mo_orderpack::sort_index`
- interface `mo_orderpack::ctrper`
- interface `mo_orderpack::fndnth`
- interface `mo_orderpack::indmed`
- interface `mo_orderpack::indnth`
- interface `mo_orderpack::inspar`
- interface `mo_orderpack::inssor`
- interface `mo_orderpack::omedian`
- interface `mo_orderpack::mrgref`
- interface `mo_orderpack::mrgrnk`
- interface `mo_orderpack::mulcnt`
- interface `mo_orderpack::rapknr`
- interface `mo_orderpack::refpar`
- interface `mo_orderpack::refsor`
- interface `mo_orderpack::rinpar`
- interface `mo_orderpack::rnkpar`
- interface `mo_orderpack::uniinv`
- interface `mo_orderpack::nearless`
- interface `mo_orderpack::unipar`
- interface `mo_orderpack::unirnk`
- interface `mo_orderpack::unista`
- interface `mo_orderpack::valmed`
- interface `mo_orderpack::valnth`

Modules

- module `mo_orderpack`
Sort and ranking routines.

Functions/Subroutines

- integer(i4) function, dimension(size(arr)) `mo_orderpack::sort_index_dp` (arr)
- integer(i4) function, dimension(size(arr)) `mo_orderpack::sort_index_sp` (arr)
- integer(i4) function, dimension(size(arr)) `mo_orderpack::sort_index_i4` (arr)
- subroutine, private `mo_orderpack::d_ctrper` (XDONT, PCLS)
- subroutine, private `mo_orderpack::r_ctrper` (XDONT, PCLS)
- subroutine, private `mo_orderpack::i_ctrper` (XDONT, PCLS)
- real(kind=dp) function, private `mo_orderpack::d_fndnth` (XDONT, NORD)
- real(kind=sp) function, private `mo_orderpack::r_fndnth` (XDONT, NORD)
- integer(kind=i4) function, private `mo_orderpack::i_fndnth` (XDONT, NORD)
- subroutine, private `mo_orderpack::d_indmed` (XDONT, INDM)
- recursive subroutine, private `mo_orderpack::d_med` (XDATT, IDATT, ires_med)
- subroutine, private `mo_orderpack::r_indmed` (XDONT, INDM)
- recursive subroutine, private `mo_orderpack::r_med` (XDATT, IDATT, ires_med)
- subroutine, private `mo_orderpack::i_indmed` (XDONT, INDM)
- recursive subroutine, private `mo_orderpack::i_med` (XDATT, IDATT, ires_med)
- integer(kind=i4) function, private `mo_orderpack::d_innth` (XDONT, NORD)

- integer(kind=i4) function, private `mo_orderpack::r_indnth` (XDONT, NORD)
- integer(kind=i4) function, private `mo_orderpack::i_indnth` (XDONT, NORD)
- subroutine, private `mo_orderpack::d_inspar` (XDONT, NORD)
- subroutine, private `mo_orderpack::r_inspar` (XDONT, NORD)
- subroutine, private `mo_orderpack::i_inspar` (XDONT, NORD)
- subroutine, private `mo_orderpack::d_inssor` (XDONT)
- subroutine, private `mo_orderpack::r_inssor` (XDONT)
- subroutine, private `mo_orderpack::i_inssor` (XDONT)
- real(kind=dp) function, private `mo_orderpack::d_median` (XDONT)
- real(kind=sp) function, private `mo_orderpack::r_median` (XDONT)
- integer(kind=i4) function, private `mo_orderpack::i_median` (XDONT)
- subroutine, private `mo_orderpack::d_mrgref` (XVALT, IRNGT)
- subroutine, private `mo_orderpack::r_mrgref` (XVALT, IRNGT)
- subroutine, private `mo_orderpack::i_mrgref` (XVALT, IRNGT)
- subroutine, private `mo_orderpack::d_mrgrnk` (XDONT, IRNGT)
- subroutine, private `mo_orderpack::r_mrgrnk` (XDONT, IRNGT)
- subroutine, private `mo_orderpack::i_mrgrnk` (XDONT, IRNGT)
- subroutine, private `mo_orderpack::d_mulcnt` (XDONT, IMULT)
- subroutine, private `mo_orderpack::r_mulcnt` (XDONT, IMULT)
- subroutine, private `mo_orderpack::i_mulcnt` (XDONT, IMULT)
- subroutine, private `mo_orderpack::d_rapknr` (XDONT, IRNGT, NORD)
- subroutine, private `mo_orderpack::r_rapknr` (XDONT, IRNGT, NORD)
- subroutine, private `mo_orderpack::i_rapknr` (XDONT, IRNGT, NORD)
- subroutine, private `mo_orderpack::d_refpar` (XDONT, IRNGT, NORD)
- subroutine, private `mo_orderpack::r_refpar` (XDONT, IRNGT, NORD)
- subroutine, private `mo_orderpack::i_refpar` (XDONT, IRNGT, NORD)
- subroutine, private `mo_orderpack::d_refsor` (XDONT)
- recursive subroutine, private `mo_orderpack::d_substor` (XDONT, IDEB1, IFIN1)
- subroutine, private `mo_orderpack::r_refsor` (XDONT)
- recursive subroutine, private `mo_orderpack::r_substor` (XDONT, IDEB1, IFIN1)
- subroutine, private `mo_orderpack::i_refsor` (XDONT)
- recursive subroutine, private `mo_orderpack::i_substor` (XDONT, IDEB1, IFIN1)
- subroutine, private `mo_orderpack::d_rinpar` (XDONT, IRNGT, NORD)
- subroutine, private `mo_orderpack::r_rinpar` (XDONT, IRNGT, NORD)
- subroutine, private `mo_orderpack::i_rinpar` (XDONT, IRNGT, NORD)
- subroutine, private `mo_orderpack::d_rnkpar` (XDONT, IRNGT, NORD)
- subroutine, private `mo_orderpack::r_rnkpar` (XDONT, IRNGT, NORD)
- subroutine, private `mo_orderpack::i_rnkpar` (XDONT, IRNGT, NORD)
- subroutine, private `mo_orderpack::d_uniinv` (XDONT, IGOEST)
- subroutine, private `mo_orderpack::r_uniinv` (XDONT, IGOEST)
- subroutine, private `mo_orderpack::i_uniinv` (XDONT, IGOEST)
- real(kind=dp) function, private `mo_orderpack::d_nearless` (XVAL)
- real(kind=sp) function, private `mo_orderpack::r_nearless` (XVAL)
- integer(kind=i4) function, private `mo_orderpack::i_nearless` (XVAL)
- subroutine, private `mo_orderpack::d_unipar` (XDONT, IRNGT, NORD)
- subroutine, private `mo_orderpack::r_unipar` (XDONT, IRNGT, NORD)
- subroutine, private `mo_orderpack::i_unipar` (XDONT, IRNGT, NORD)
- subroutine, private `mo_orderpack::d_unirnk` (XVALT, IRNGT, NUNI)
- subroutine, private `mo_orderpack::r_unirnk` (XVALT, IRNGT, NUNI)
- subroutine, private `mo_orderpack::i_unirnk` (XVALT, IRNGT, NUNI)
- subroutine, private `mo_orderpack::d_unista` (XDONT, NUNI)
- subroutine, private `mo_orderpack::r_unista` (XDONT, NUNI)
- subroutine, private `mo_orderpack::i_unista` (XDONT, NUNI)
- recursive real(kind=dp) function, private `mo_orderpack::d_valmed` (XDONT)
- recursive real(kind=sp) function, private `mo_orderpack::r_valmed` (XDONT)

- recursive integer(kind=i4) function, private [mo_orderpack::i_valmed](#) (XDONT)
- real(kind=dp) function, private [mo_orderpack::d_valnth](#) (XDONT, NORD)
- real(kind=sp) function, private [mo_orderpack::r_valnth](#) (XDONT, NORD)
- integer(kind=i4) function, private [mo_orderpack::i_valnth](#) (XDONT, NORD)

Variables

- integer(kind=i4), dimension(:), allocatable, save [mo_orderpack::idont](#)

17.81 mo_percentile.f90 File Reference

Data Types

- interface [mo_percentile::median](#)
- interface [mo_percentile::n_element](#)
- interface [mo_percentile::percentile](#)
- interface [mo_percentile::qmedian](#)

Modules

- module [mo_percentile](#)

Functions/Subroutines

- real(dp) function [mo_percentile::median_dp](#) (arrin, mask)
- real(sp) function [mo_percentile::median_sp](#) (arrin, mask)
- real(dp) function [mo_percentile::n_element_dp](#) (idat, n, mask, before, after, previous, next)
- real(sp) function [mo_percentile::n_element_sp](#) (idat, n, mask, before, after, previous, next)
- real(dp) function [mo_percentile::percentile_0d_dp](#) (arrin, k, mask, mode_in)
- real(sp) function [mo_percentile::percentile_0d_sp](#) (arrin, k, mask, mode_in)
- real(dp) function, dimension(size(k)) [mo_percentile::percentile_1d_dp](#) (arrin, k, mask, mode_in)
- real(sp) function, dimension(size(k)) [mo_percentile::percentile_1d_sp](#) (arrin, k, mask, mode_in)
- real(dp) function [mo_percentile::qmedian_dp](#) (dat)
- real(sp) function [mo_percentile::qmedian_sp](#) (dat)

17.82 mo_pet.f90 File Reference

Modules

- module [mo_pet](#)

Module for calculating reference/potential evapotranspiration [mm s-1].

Functions/Subroutines

- elemental pure real(dp) function, public [mo_pet::pet_hargreaves](#) (HarSamCoeff, HarSamConst, tavg, tmax, tmin, latitude, doy)

Reference Evapotranspiration after Hargreaves.

- elemental pure real(dp) function, public [mo_pet::pet_priestly](#) (PrieTayParam, Rn, tavg)

Reference Evapotranspiration after Priestly-Taylor.

- elemental pure real(dp) function, public `mo_pet::pet_penman` (net_rad, tavg, act_vap_pressure, aerodyn← resistance, bulksurface_resistance, a_s, a_sh)
Reference Evapotranspiration after Penman-Monteith.
- elemental pure real(dp) function, private `mo_pet::extraterr_rad_approx` (doy, latitude)
Approximation of extraterrestrial radiation.
- elemental pure real(dp) function, private `mo_pet::slope_satpressure` (tavg)
slope of saturation vapour pressure curve
- elemental pure real(dp) function, private `mo_pet::sat_vap_pressure` (tavg)
calculation of the saturation vapour pressure

17.83 `mo_prepare_gridded_lai.f90` File Reference

Modules

- module `mo_prepare_gridded_lai`
Prepare daily LAI fields (e.g., MODIS data) for mHM.

Functions/Subroutines

- subroutine, public `mo_prepare_gridded_lai::prepare_gridded_daily_lai_data` (iBasin, nRows, nCols, mask, L← AIPer_iBasin)
Prepare gridded daily LAI data.
- subroutine, public `mo_prepare_gridded_lai::prepare_gridded_mean_monthly_lai_data` (iBasin, nRows, nCols, mask)
prepare_gridded_mean_monthly_LAI_data

17.84 `mo_read_forcing_nc.f90` File Reference

Modules

- module `mo_read_forcing_nc`
Reads forcing input data.

Functions/Subroutines

- subroutine, public `mo_read_forcing_nc::read_forcing_nc` (folder, nRows, nCols, varName, mask, data, target_period, lower, upper, nctimestep, fileName, nocheck, maskout)
Reads forcing input in NetCDF file format.
- subroutine, public `mo_read_forcing_nc::read_weights_nc` (folder, nRows, nCols, varName, data, mask, lower, upper, nocheck, maskout, fileName)
Reads weights for meteo forcings input in NetCDF file format.
- subroutine `mo_read_forcing_nc::get_time_vector_and_select` (var, fname, inctimestep, time_start, time_cnt, target_period)
TODO: add description.

17.85 mo_read_latlon.f90 File Reference

Modules

- module [mo_read_latlon](#)
reading latitude and longitude coordinates for each basin

Functions/Subroutines

- subroutine, public [mo_read_latlon::read_latlon](#) (ii, lon_var_name, lat_var_name, level_name, level)
reads latitude and longitude coordinates

17.86 mo_read_lut.f90 File Reference

Modules

- module [mo_read_lut](#)
Routines reading lookup tables (lut).

Functions/Subroutines

- subroutine, public [mo_read_lut::read_geoformation_lut](#) (filename, fileunit, nGeo, geo_unit, geo_karstic)
Reads LUT containing geological formation information.
- subroutine, public [mo_read_lut::read_lai_lut](#) (filename, fileunit, nLAI, LAIIDlist, LAI)
Reads LUT containing LAI information.

17.87 mo_read_optional_data.f90 File Reference

Modules

- module [mo_read_optional_data](#)
Read optional data for mHM calibration.

Functions/Subroutines

- subroutine, public [mo_read_optional_data::read_soil_moisture](#) (iBasin)
Read soil moisture data from NetCDF file for calibration.
- subroutine, public [mo_read_optional_data::read_basin_avg_tws](#)
Read basin average TWS timeseries from file, the same way runoff is read.
- subroutine, public [mo_read_optional_data::read_neutrons](#) (iBasin)
Read neutrons data from NetCDF file for calibration.
- subroutine, public [mo_read_optional_data::read_evapotranspiration](#) (iBasin)
Read evapotranspiration data from NetCDF file for calibration.

17.88 mo_read_spatial_data.f90 File Reference

Data Types

- interface `mo_read_spatial_data::read_spatial_data_ascii`
Reads spatial data files of ASCII format.

Modules

- module `mo_read_spatial_data`
Reads spatial input data.

Functions/Subroutines

- subroutine `mo_read_spatial_data::read_spatial_data_ascii_dp` (filename, fileunit, header_ncols, header_nrows, header_xllcorner, header_yllcorner, header_cellsize, data, mask)
TODO: add description.
- subroutine `mo_read_spatial_data::read_spatial_data_ascii_i4` (filename, fileunit, header_ncols, header_nrows, header_xllcorner, header_yllcorner, header_cellsize, data, mask)
TODO: add description.
- subroutine, public `mo_read_spatial_data::read_header_ascii` (filename, fileunit, header_ncols, header_nrows, header_xllcorner, header_yllcorner, header_cellsize, header_nodata)
Reads header lines of ASCII files.

17.89 mo_read_timeseries.f90 File Reference

Modules

- module `mo_read_timeseries`
Routines to read files containing timeseries data.

Functions/Subroutines

- subroutine, public `mo_read_timeseries::read_timeseries` (filename, fileunit, periodStart, periodEnd, optimize, opti_function, data, mask, nMeasPerDay)
Reads time series in ASCII format.

17.90 mo_read_wrapper.f90 File Reference

Modules

- module `mo_read_wrapper`
Wrapper for all reading routines.

Functions/Subroutines

- subroutine, public `mo_read_wrapper::read_data` (LAIPer)
Reads data.
- subroutine `mo_read_wrapper::check_consistency_lut_map` (data, lookuptable, filename, unique_values)
Checks if classes in input maps appear in look up tables.

17.91 mo_restart.f90 File Reference

Data Types

- interface `mo_restart::unpack_field_and_write`
TODO: add description.

Modules

- module `mo_restart`
reading and writing states, fluxes and configuration for restart of mHM.

Functions/Subroutines

- subroutine, public `mo_restart::write_restart_files` (OutPath)
write restart files for each basin
- subroutine, public `mo_restart::read_restart_states` (iBasin, InPath)
reads fluxes and state variables from file
- subroutine `mo_restart::unpack_field_and_write_1d_i4` (nc, var_name, var_dims, fill_value, data, mask, var←_long_name)
- subroutine `mo_restart::unpack_field_and_write_1d_dp` (nc, var_name, var_dims, fill_value, data, mask, var←_long_name)
- subroutine `mo_restart::unpack_field_and_write_2d_dp` (nc, var_name, var_dims, fill_value, data, mask, var←_long_name)
- subroutine `mo_restart::unpack_field_and_write_3d_dp` (nc, var_name, var_dims, fill_value, data, mask, var←_long_name)

17.92 mo_runoff.f90 File Reference

Modules

- module `mo_runoff`
Runoff generation for the unsaturated zone, saturated zone (or groundwater zone), and runoff accumulation.

Functions/Subroutines

- subroutine, public `mo_runoff::runoff_unsat_zone` (k1, kp, k0, alpha, karst_loss, pefec_soil, unsat_thresh, sat_storage, unsat_storage, slow_interflow, fast_interflow, perc)
Runoff generation for the saturated zone.
- subroutine, public `mo_runoff::runoff_sat_zone` (k2, sat_storage, baseflow)
Runoff generation for the saturated zone.
- subroutine, public `mo_runoff::l1_total_runoff` (fSealed_area_fraction, fast_interflow, slow_interflow, baseflow, direct_runoff, total_runoff)
total runoff accumulation at level 1

17.93 mo_sce.f90 File Reference

Modules

- module `mo_sce`
Shuffled Complex Evolution optimization algorithm.

Functions/Subroutines

- real(dp) function, dimension(size(pini, 1)), public `mo_sce::sce` (eval, functn, pini, prange, mymaxn, mymaxit, mykstop, mypcento, mypeps, myseed, myngs, mynpg, mynps, mynspl, mymings, myiniflg, myprint, mymask, myalpha, mybeta, tmp_file, popul_file, popul_file_append, parallel, restart, restart_file, bestf, neval, history)

Shuffled Complex Evolution (SCE) algorithm for global optimization.

- subroutine `write_best_intermediate` (to_file)
- subroutine `write_best_final` ()
- subroutine `write_population` (to_file)
- subroutine `write_termination_case` (case)
- subroutine `set_optional` ()
- subroutine `mo_sce::parsst` (x, bound, peps, mask, xnstd, gnrng, ipcnvg)
- subroutine `mo_sce::comp` (ngs2, npg, a, af, b, bf)
- subroutine `mo_sce::sort_matrix` (rb, ra)
- subroutine `mo_sce::chkcst` (x, bl, bu, mask, ibound)
- subroutine `mo_sce::getpnt` (idist, bl, bu, std, xi, mask, save_state, x)
- subroutine `mo_sce::cce` (s, sf, bl, bu, maskpara, xnstd, icall, maxn, maxit, save_state_gauss, functn, eval, alpha, beta, history, idot)

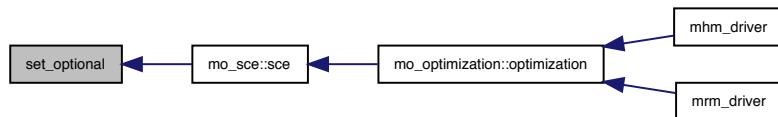
17.93.1 Function/Subroutine Documentation

17.93.1.1 `set_optional()`

```
subroutine sce::set_optional ( )  [private]
```

Referenced by `mo_sce::sce()`.

Here is the caller graph for this function:

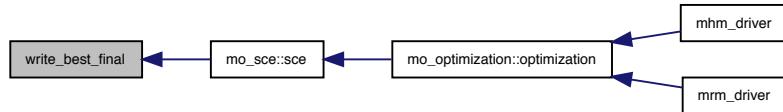


17.93.1.2 `write_best_final()`

```
subroutine sce::write_best_final ( )  [private]
```

Referenced by `mo_sce::sce()`.

Here is the caller graph for this function:

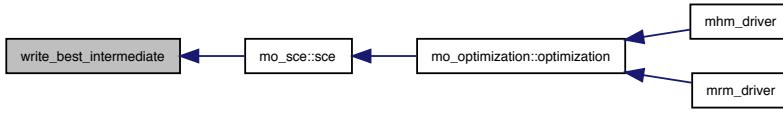


17.93.1.3 write_best_intermediate()

```
subroutine sce::write_best_intermediate (
    logical, intent(in) to_file )
```

Referenced by mo_sce::sce().

Here is the caller graph for this function:

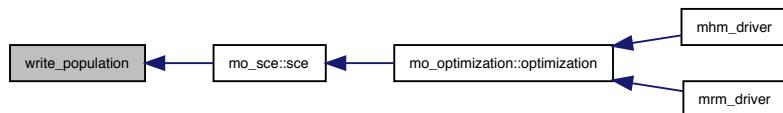


17.93.1.4 write_population()

```
subroutine sce::write_population (
    logical, intent(in) to_file ) [private]
```

Referenced by mo_sce::sce().

Here is the caller graph for this function:

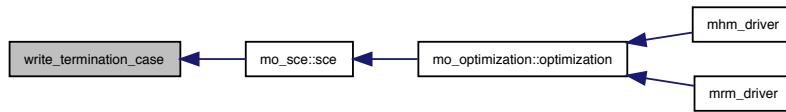


17.93.1.5 write_termination_case()

```
subroutine sce::write_termination_case (
    integer(i4), intent(in) case ) [private]
```

Referenced by mo_sce::sce().

Here is the caller graph for this function:



17.94 mo_set_netcdf_outputs.f90 File Reference

Modules

- module [mo_set_netcdf_outputs](#)
Defines the structure of the netCDF to write the output in.

Functions/Subroutines

- subroutine, public [mo_set_netcdf_outputs::set_netcdf](#) (NoNetcdfVars, nrows, ncols)
Runs mhm with a specific parameter set and returns required variables, e.g. runoff.

17.95 mo_snow_accum_melt.f90 File Reference

Modules

- module [mo_snow_accum_melt](#)
Snow melting and accumulation.

Functions/Subroutines

- subroutine, public [mo_snow_accum_melt::snow_accum_melt](#) (deg_day_incr, deg_day_max, deg_day_←
noprec, prec, temperature, temperature_thresh, thrfall, snow_pack, deg_day, melt, prec_effect, rain, snow)
Snow melting and accumulation.

17.96 mo_soil_database.f90 File Reference

Modules

- module [mo_soil_database](#)
Generating soil database from input file.

Functions/Subroutines

- subroutine, public `mo_soil_database::read_soil_lut` (filename)
Reads the soil LUT file.
- subroutine, public `mo_soil_database::generate_soil_database`
Generates soil database.

17.97 mo_soil_moisture.f90 File Reference

Modules

- module `mo_soil_moisture`
Soil moisture of the different layers.

Functions/Subroutines

- subroutine, public `mo_soil_moisture::soil_moisture` (processCase, frac_sealed, water_thresh_sealed, pet, evap_coeff, soil_moist_sat, frac_roots, soil_moist_FC, wilting_point, soil_moist_exponen, jarvis_thresh_c1, aet_canopy, prec_effec, runoff_sealed, storage_sealed, infiltration, soil_moist, aet, aet_sealed)
Soil moisture in different soil horizons.
- elemental pure real(dp) function, private `mo_soil_moisture::feddes_et_reduction` (soil_moist, soil_moist_FC, wilting_point, frac_roots)
stress factor for reducing evapotranspiration based on actual soil moisture
- elemental pure real(dp) function, private `mo_soil_moisture::jarvis_et_reduction` (soil_moist, soil_moist_sat, wilting_point, frac_roots, jarvis_thresh_c1)
stress factor for reducing evapotranspiration based on actual soil moisture

17.98 mo_spatial_agg_disagg_forcing.f90 File Reference

Data Types

- interface `mo_spatial_agg_disagg_forcing::spatial_aggregation`
Spatial aggregation of meterological variables.
- interface `mo_spatial_agg_disagg_forcing::spatial_disaggregation`
Spatial disaggregation of meterological variables.

Modules

- module `mo_spatial_agg_disagg_forcing`
Spatial aggregation or disaggregation of meteorological input data.

Functions/Subroutines

- subroutine `mo_spatial_agg_disagg_forcing::spatial_aggregation_3d` (data2, cellsize2, cellsize1, mask1, mask2, data1)
- subroutine `mo_spatial_agg_disagg_forcing::spatial_aggregation_4d` (data2, cellsize2, cellsize1, mask1, mask2, data1)
- subroutine `mo_spatial_agg_disagg_forcing::spatial_disaggregation_3d` (data2, cellsize2, cellsize1, mask1, mask2, data1)
- subroutine `mo_spatial_agg_disagg_forcing::spatial_disaggregation_4d` (data2, cellsize2, cellsize1, mask1, mask2, data1)

17.99 mo_spatialsimilarity.f90 File Reference

Data Types

- interface [mo_spatialsimilarity::nndv](#)
Calculates the number of neighboring dominating values, a measure for spatial dissimilarity.
- interface [mo_spatialsimilarity::pd](#)
Calculates pattern dissimilarity (PD) measure.

Modules

- module [mo_spatialsimilarity](#)
Routines for bias insensitive comparison of spatial patterns.

Functions/Subroutines

- real(sp) function [mo_spatialsimilarity::nndv_sp](#) (mat1, mat2, mask, valid)
- real(dp) function [mo_spatialsimilarity::nndv_dp](#) (mat1, mat2, mask, valid)
- real(sp) function [mo_spatialsimilarity::pd_sp](#) (mat1, mat2, mask, valid)
- real(dp) function [mo_spatialsimilarity::pd_dp](#) (mat1, mat2, mask, valid)

17.100 mo_standard_score.f90 File Reference

Data Types

- interface [mo_standard_score::standard_score](#)
Calculates the standard score / normalization (anomaly) / z-score.
- interface [mo_standard_score::classified_standard_score](#)
Calculates the classified standard score (e.g. classes are months).

Modules

- module [mo_standard_score](#)
Routines for calculating the normalization (anomaly)/standard score/z score and the deseasonalized (standard score on monthly basis) values of a time series.

Functions/Subroutines

- real(sp) function, dimension(size(data, dim=1)) [mo_standard_score::standard_score_sp](#) (data, mask)
- real(dp) function, dimension(size(data, dim=1)) [mo_standard_score::standard_score_dp](#) (data, mask)
- real(sp) function, dimension(size(data, dim=1)) [mo_standard_score::classified_standard_score_sp](#) (data, classes, mask)
- real(dp) function, dimension(size(data, dim=1)) [mo_standard_score::classified_standard_score_dp](#) (data, classes, mask)

17.101 mo_startup.f90 File Reference

Modules

- module [mo_startup](#)
Startup procedures for mHM.

Functions/Subroutines

- subroutine, public `mo_startup::mhm_initialize`
Initialize main mHM variables.
- subroutine `mo_startup::constants_init`
Initialize mHM constants.
- subroutine `mo_startup::l2_variable_init` (iBasin, level0_iBasin, level2_iBasin)
Initialize Level-2 meteorological forcings data.

17.102 mo_string_utils.f90 File Reference

Data Types

- interface `mo_string_utils::num2str`
Convert to string.
- interface `mo_string_utils::numarray2str`
Convert to string.

Modules

- module `mo_string_utils`
String utilities.

Functions/Subroutines

- character(len(whitespaces)) function, public `mo_string_utils::compress` (whiteSpaces, n)
- subroutine, public `mo_string_utils::divide_string` (string, delim, strArr)
Divide string in substrings.
- logical function, public `mo_string_utils::equalstrings` (string1, string2)
- logical function, public `mo_string_utils::nonull` (str)
Checks if string was already used.
- character(len=256) function, dimension(:,), allocatable, public `mo_string_utils::splitstring` (string, delim)
- logical function, public `mo_string_utils::startswith` (string, start)
- character(len=len_trim(upper)) function, public `mo_string_utils::tolower` (upper)
Convert to lower case.
- character(len=len_trim(lower)) function, public `mo_string_utils::toupper` (lower)
- pure character(len=10) function `mo_string_utils::i42str` (nn, form)
- pure character(len=20) function `mo_string_utils::i82str` (nn, form)
- pure character(len=32) function `mo_string_utils::sp2str` (rr, form)
- pure character(len=32) function `mo_string_utils::dp2str` (rr, form)
- pure character(len=10) function `mo_string_utils::log2str` (ll, form)
- character(len=size(arr)) function `mo_string_utils::i4array2str` (arr)
- integer(i4) function, dimension(:,), allocatable, public `mo_string_utils::str2num` (string)

Variables

- character(len=*), parameter, public `mo_string_utils::separator` = repeat('-', 70)

17.103 mo_template.f90 File Reference

Data Types

- interface `mo_template::mean`

The average.

Modules

- module `mo_template`

Template for future module developments.

Functions/Subroutines

- elemental pure real(dp) function, public `mo_template::circum` (radius)

Circumference of a circle.

- real(dp) function `mo_template::mean_dp` (dat, mask)

- real(sp) function `mo_template::mean_sp` (dat, mask)

Variables

- real(dp), parameter, public `mo_template::pi_dp` = 3.141592653589793238462643383279502884197_dp

Constant Pi in double precision.

- real(sp), parameter, public `mo_template::pi_sp` = 3.141592653589793238462643383279502884197_sp

Constant Pi in single precision.

- integer(i4), parameter `mo_template::itest` = 1

17.104 mo_temporal_aggregation.f90 File Reference

Data Types

- interface `mo_temporal_aggregation::day2mon_average`

Day-to-month average (`day2mon_average`)

- interface `mo_temporal_aggregation::hour2day_average`

Hour-to-day average (`hour2day_average`)

Modules

- module `mo_temporal_aggregation`

Temporal aggregation for time series (averaging)

Functions/Subroutines

- subroutine `mo_temporal_aggregation::day2mon_average_dp` (daily_data, yearS, monthS, dayS, mon_avg, misval, rm_misval)

- subroutine `mo_temporal_aggregation::hour2day_average_dp` (hourly_data, yearS, monthS, dayS, hourS, day_avg, misval, rm_misval)

17.105 mo_temporal_disagg_forcing.f90 File Reference

Modules

- module [mo_temporal_disagg_forcing](#)
Temporal disaggregation of daily input values.

Functions/Subroutines

- elemental pure subroutine, public [mo_temporal_disagg_forcing::temporal_disagg_forcing](#) (isday, ntimesteps←_day, prec_day, pet_day, temp_day, fday_prec, fday_pet, fday_temp, fnight_prec, fnight_pet, fnight_temp, temp_weights, pet_weights, pre_weights, read_meteo_weights, prec, pet, temp)
Temporally distribute daily mean forcings onto time step.

17.106 mo_timer.f90 File Reference

Modules

- module [mo_timer](#)
Timing routines.

Functions/Subroutines

- subroutine, public [mo_timer::timer_check](#) (timer)
Check a timer.
- subroutine, public [mo_timer::timer_clear](#) (timer)
Reset a timer.
- real(sp) function, public [mo_timer::timer_get](#) (timer)
Return a timer.
- subroutine, public [mo_timer::timer_print](#) (timer)
Print a timer.
- subroutine, public [mo_timer::timer_start](#) (timer)
Start a timer.
- subroutine, public [mo_timer::timer_stop](#) (timer)
Stop a timer.
- subroutine, public [mo_timer::timers_init](#)
Initialise timer module.

Variables

- integer(i4), parameter, public [mo_timer::max_timers](#) = 99
max number of timers allowed
- integer(i4), save, public [mo_timer::cycles_max](#)
max value of clock allowed by system
- real(sp), save, public [mo_timer::clock_rate](#)
clock_rate in seconds for each cycle
- integer(i4), dimension(max_timers), save, public [mo_timer::cycles1](#)
cycle number at start for each timer
- integer(i4), dimension(max_timers), save, public [mo_timer::cycles2](#)

- **real(sp)**, dimension(max_timers), save, public **mo_timer::cputime**
accumulated cpu time in each timer
- **character(len=8)**, dimension(max_timers), save, public **mo_timer::status**
timer status string

17.107 mo_upscaling_operators.f90 File Reference

Modules

- module **mo_upscaling_operators**
Module containing upscaling operators.

Functions/Subroutines

- **integer(i4)** function, dimension(size(l1_upper_rowid_cell, 1)), public **mo_upscaling_operators::majority_statistics** (nClass, L1_upper_rowId_cell, L1_lower_rowId_cell, L1_left_colonId_cell, L1_right_colonId_cell, L0_fineScale_2D_data)
majority statistics
- **real(dp)** function, dimension(size(l0upbound_inlx, 1)), public **mo_upscaling_operators::l0_fractionalcover_in_lx** (dataIn0, classId, mask0, L0upBound_inLx, L0downBound_inLx, L0leftBound_inLx, L0rightBound_inLx, nTCells0_inLx)
fractional coverage of a given class of L0 fields in Lx field (Lx = L1 or L11)
- **real(dp)** function, dimension(size(nl0_cells_in_l1_cell, 1)), public **mo_upscaling_operators::upscale_arithmetic_mean** (nl0_cells_in_L1_cell, L1_upper_rowId_cell, L1_lower_rowId_cell, L1_left_colonId_cell, L1_right_colonId_cell, L0_cellId, mask0, nodata_value, L0_fineScale_data)
arithmetic mean
- **real(dp)** function, dimension(size(nl0_cells_in_l1_cell, 1)), public **mo_upscaling_operators::upscale_harmonic_mean** (nl0_cells_in_L1_cell, L1_upper_rowId_cell, L1_lower_rowId_cell, L1_left_colonId_cell, L1_right_colonId_cell, L0_cellId, mask0, nodata_value, L0_fineScale_data)
harmonic mean
- **real(dp)** function, dimension(size(l1_upper_rowid_cell, 1)), public **mo_upscaling_operators::upscale_geometric_mean** (L1_upper_rowId_cell, L1_lower_rowId_cell, L1_left_colonId_cell, L1_right_colonId_cell, mask0, nodata_value, L0_fineScale_data)
geometric mean
- **real(dp)** function, dimension(size(nl0_cells_in_l1_cell, 1)) **mo_upscaling_operators::upscale_p_norm** (nl0_cells_in_L1_cell, L1_upper_rowId_cell, L1_lower_rowId_cell, L1_left_colonId_cell, L1_right_colonId_cell, L0_cellId, mask0, nodata_value, p_norm, L0_fineScale_data)
arithmetic mean

17.108 mo_utils.f90 File Reference

Data Types

- interface **mo_utils::equal**
Comparison of real values.
- interface **mo_utils::notequal**
- interface **mo_utils::greaterequal**
- interface **mo_utils::lesserequal**
- interface **mo_utils::eq**
- interface **mo_utils::ne**

- interface `mo_utils::ge`
- interface `mo_utils::le`
- interface `mo_utils::is_finite`
 $.true.$ if not IEEE Inf, IEEE NaN, nor IEEE Inf nor IEEE NaN, respectively.
- interface `mo_utils::is_nan`
- interface `mo_utils::is_normal`
- interface `mo_utils::locate`
 $Find$ closest values in a monotonic series, returns the indexes.
- interface `mo_utils::swap`
 $Swap$ to values or two elements in array.
- interface `mo_utils::special_value`
 $Special$ IEEE values.

Modules

- module `mo_utils`
General utilities for the CHS library.

Functions/Subroutines

- elemental pure logical function `mo_utils::equal_dp` (a, b)
- elemental pure logical function `mo_utils::equal_sp` (a, b)
- elemental pure logical function `mo_utils::greaterequal_dp` (a, b)
- elemental pure logical function `mo_utils::greaterequal_sp` (a, b)
- elemental pure logical function `mo_utils::lesserequal_dp` (a, b)
- elemental pure logical function `mo_utils::lesserequal_sp` (a, b)
- elemental pure logical function `mo_utils::notequal_dp` (a, b)
- elemental pure logical function `mo_utils::notequal_sp` (a, b)
- elemental pure logical function `mo_utils::is_finite_dp` (a)
- elemental pure logical function `mo_utils::is_finite_sp` (a)
- elemental pure logical function `mo_utils::is_nan_dp` (a)
- elemental pure logical function `mo_utils::is_nan_sp` (a)
- elemental pure logical function `mo_utils::is_normal_dp` (a)
- elemental pure logical function `mo_utils::is_normal_sp` (a)
- integer(i4) function `mo_utils::locate_0d_dp` (x, y)
- integer(i4) function `mo_utils::locate_0d_sp` (x, y)
- integer(i4) function, dimension(:), allocatable `mo_utils::locate_1d_dp` (x, y)
- integer(i4) function, dimension(:), allocatable `mo_utils::locate_1d_sp` (x, y)
- elemental pure subroutine `mo_utils::swap_xy_dp` (x, y)
- elemental pure subroutine `mo_utils::swap_xy_sp` (x, y)
- elemental pure subroutine `mo_utils::swap_xy_i4` (x, y)
- subroutine `mo_utils::swap_vec_dp` (x, i1, i2)
- subroutine `mo_utils::swap_vec_sp` (x, i1, i2)
- subroutine `mo_utils::swap_vec_i4` (x, i1, i2)
- real(dp) function `mo_utils::special_value_dp` (x, ieee)
- real(sp) function `mo_utils::special_value_sp` (x, ieee)

17.109 mo_write_ascii.f90 File Reference

Modules

- module `mo_write_ascii`
Module to write ascii file output.

Functions/Subroutines

- subroutine, public `mo_write_ascii::write_configfile`

This module writes the results of the configuration into an ASCII-file.
- subroutine, public `mo_write_ascii::write_optifile` (best_OF, best_paramSet, param_names)

Write briefly final optimization results.
- subroutine, public `mo_write_ascii::write_optinamelist` (processMatrix, parameters, maskpara, parameters_← name)

Write final, optimized parameter set in a namelist format.

17.110 `mo_write_fluxes_states.f90` File Reference

Data Types

- interface `mo_write_fluxes_states::outputvariable`
- interface `mo_write_fluxes_states::outputvariable`
- interface `mo_write_fluxes_states::outputdataset`
- interface `mo_write_fluxes_states::outputdataset`

Modules

- module `mo_write_fluxes_states`

Creates NetCDF output for different fluxes and state variables of mHM.

Functions/Subroutines

- type(outputvariable) function `mo_write_fluxes_states::newoutputvariable` (nc, name, dtype, dims, ncells, mask, avg)

Initialize OutputVariable.
- subroutine `mo_write_fluxes_states::updatevariable` (self, data)

Update OutputVariable.
- subroutine `mo_write_fluxes_states::writevariabletimestep` (self, timestep)

Write timestep to file.
- type(outputdataset) function, public `mo_write_fluxes_states::newoutputdataset` (ibasin, mask1, nCells)

Initialize OutputDataset.
- subroutine `mo_write_fluxes_states::updatedataset` (self, sidx, eidx, L1_fSealed, L1_fNotSealed, L1_inter, L1_snowPack, L1_soilMoist, L1_soilMoistSat, L1_sealSTW, L1_unsatSTW, L1_satSTW, L1_neutrons, L1_← pet, L1_aETSoil, L1_aETCanopy, L1_aETSealed, L1_total_runoff, L1_runoffSeal, L1_fastRunoff, L1_slow← Runoff, L1_baseflow, L1_percol, L1_infilSoil, L1_preEffect)

Update all variables.
- subroutine `mo_write_fluxes_states::writetimestep` (self, timestep)

Write all accumulated data.
- subroutine `mo_write_fluxes_states::close` (self)

Close the file.
- type(ncdataset) function `mo_write_fluxes_states::createoutputfile` (ibasin)

Create and initialize output file.
- subroutine `mo_write_fluxes_states::writevariableattributes` (var, long_name, unit)

Write output variable attributes.
- character(16) function `mo_write_fluxes_states::fluxesunit` (ibasin)

Generate a unit string.

17.111 mo_xor4096.f90 File Reference

Data Types

- interface [mo_xor4096::get_timeseed](#)
- interface [mo_xor4096::xor4096](#)
- interface [mo_xor4096::xor4096g](#)

Modules

- module [mo_xor4096](#)

Functions/Subroutines

- subroutine [mo_xor4096::get_timeseed_i4_0d](#) (seed)
- subroutine [mo_xor4096::get_timeseed_i4_1d](#) (seed)
- subroutine [mo_xor4096::get_timeseed_i8_0d](#) (seed)
- subroutine [mo_xor4096::get_timeseed_i8_1d](#) (seed)
- subroutine [mo_xor4096::xor4096s_0d](#) (seed, SingleIntegerRN, save_state)
- subroutine [mo_xor4096::xor4096s_1d](#) (seed, SingleIntegerRN, save_state)
- subroutine [mo_xor4096::xor4096f_0d](#) (seed, SingleRealRN, save_state)
- subroutine [mo_xor4096::xor4096f_1d](#) (seed, SingleRealRN, save_state)
- subroutine [mo_xor4096::xor4096l_0d](#) (seed, DoubleIntegerRN, save_state)
- subroutine [mo_xor4096::xor4096l_1d](#) (seed, DoubleIntegerRN, save_state)
- subroutine [mo_xor4096::xor4096d_0d](#) (seed, DoubleRealRN, save_state)
- subroutine [mo_xor4096::xor4096d_1d](#) (seed, DoubleRealRN, save_state)
- subroutine [mo_xor4096::xor4096gf_0d](#) (seed, SingleRealRN, save_state)
- subroutine [mo_xor4096::xor4096gf_1d](#) (seed, SingleRealRN, save_state)
- subroutine [mo_xor4096::xor4096gd_0d](#) (seed, DoubleRealRN, save_state)
- subroutine [mo_xor4096::xor4096gd_1d](#) (seed, DoubleRealRN, save_state)

Variables

- integer(i4), parameter, public [mo_xor4096::n_save_state](#) = 132_i4
Dimension of vector saving the state of a stream.

17.112 mpr_driver.f90 File Reference

Modules

- module [dummy_mpr](#)

Functions/Subroutines

- program [mpr_driver](#)
Distributed precipitation-runoff model mHM.

17.112.1 Function/Subroutine Documentation

17.112.1.1 mpr_driver()

```
program mpr_driver ( )
```

Distributed precipitation-runoff model mHM.

This is the main driver of mHM, which calls one instance of mHM for a multiple basins and a given period.

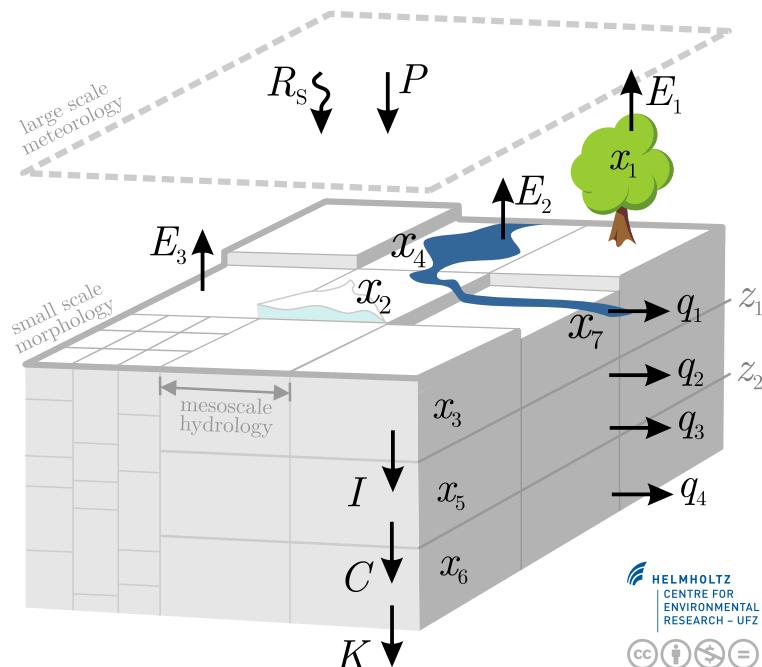


Figure 17.2: Typical mHM cell

Luis Samaniego & Rohini Kumar (UFZ)

Date

Dec 2015

Version

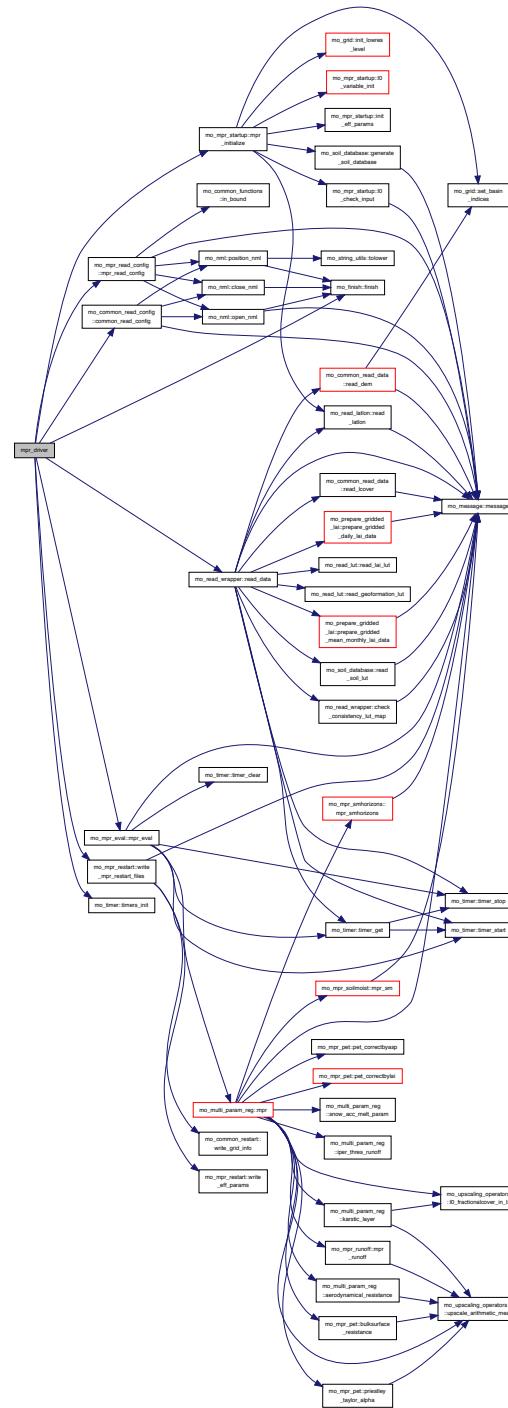
0.1

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 DEVELOPERS MAKES ANY WARRANTY, EXPRESS OR IMPLIED, OR ASSUMES ANY LIABILITY FOR THE 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.

References `mo_mpr_global_variables::c2tstu`, `mo_common_read_config::common_read_config()`, `mo_common_variables::dirrestartout`, `mo_mpr_file::file_namelist_mpr_param`, `mo_finish::finish()`, `mo_mpr_eval::mpr_eval()`, `mo_mpr_startup::mpr_initialize()`, `mo_mpr_read_config::mpr_read_config()`, `mo_read_wrapper::read_data()`, `mo_common_timer::timers_init()`, `mo_mpr_file::unamelist_mpr_param`, `mo_mpr_restart::write_mpr_restart_files()`, and `mo_common_variables::write_restart`.

Here is the call graph for this function:



Modules

- module [dummy_mrm](#)

Functions/Subroutines

- program [mrm_driver](#)

TODO: add description.

17.113.1 Function/Subroutine Documentation

17.113.1.1 [mrm_driver\(\)](#)

```
program mrm_driver ( )
```

TODO: add description.

TODO: add description

Authors

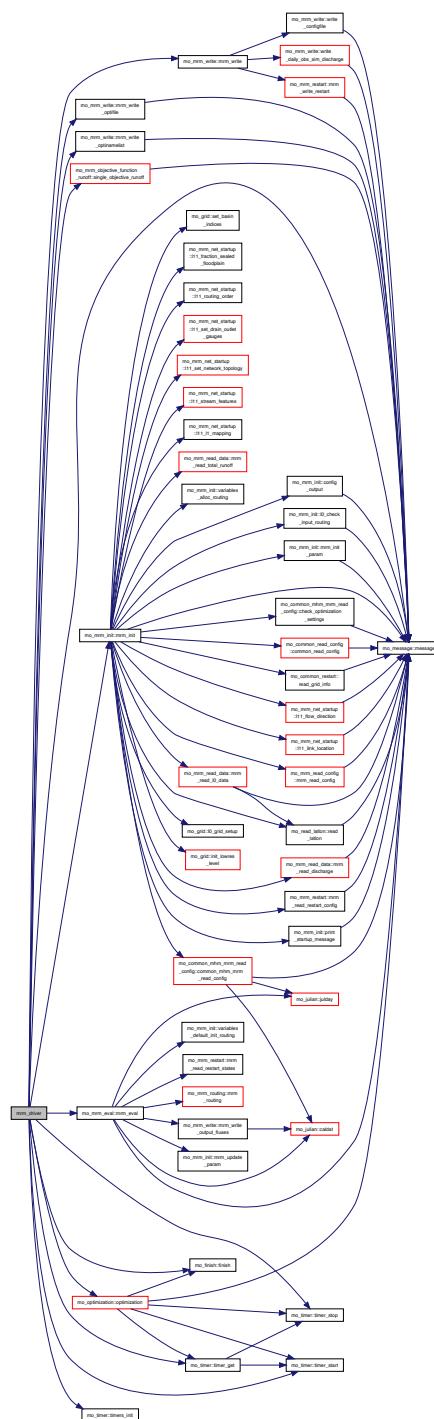
Stephan Thober

Date

Jun 2018

References `mo_common_variables::dirconfigout`, `mo_kind::dp`, `mo_mrm_file::file_namelist_mrm`, `mo_mrm_file::file_namelist_param_mrm`, `mo_finish::finish()`, `mo_common_variables::global_parameters`, `mo_common_variables::global_parameters_name`, `mo_kind::i4`, `mo_message::message()`, `mo_common_mhm_mrm_variables::mrm_coupling_mode`, `mo_mrm_eval::mrm_eval()`, `mo_mrm_init::mrm_init()`, `mo_mrm_write::mrm_write()`, `mo_mrm_write::mrm_write_optifile()`, `mo_mrm_write::mrm_write_optinamelist()`, `mo_optimization::optimization()`, `mo_common_mhm_mrm_variables::optimize`, `mo_mrm_objective_function_runoff::single_objective_runoff()`, `mo_timer::timer_get()`, `mo_timer::timer_start()`, `mo_timer::timer_stop()`, `mo_timer::timers_init()`, `mo_mrm_file::unamelist_mrm`, and `mo_mrm_file::unamelist_param_mrm`.

Here is the call graph for this function:



Bibliography

- [1] E. Aarts and J. Korst. *Simulated annealing and Boltzmann machines: a stochastic approach to combinatorial optimization and neural computing*. Wiley, Chichester, 1990. [6](#)
- [2] S. Bergström. Development and application of a conceptual runoff model for scandinavian catchments. Technical Report 7, SMHI Reports RHO, Norrköping, 1976. [1](#)
- [3] K. Beven. Prophesy, reality and uncertainty in distributed hydrological modelling. *Adv. Water Resour.*, 16:41–51, 1993. [4](#)
- [4] G. Blöschl. Scaling in hydrology. *Hydrol. Process.*, 15(4):709–711, 2001. [2](#)
- [5] G. Blöschl, C. Reszler, and J. Komma. A spatially distributed flash flood forecasting model. *Environ. Model. Soft.*, 23(4):464–478, 2008. [1](#)
- [6] G. Blöschl and M. Sivapalan. Scale issues in hydrological modelling: A review. *Hydrol. Process.*, 9(3-4):251–290, 1995. [2](#)
- [7] V. T. Chow, D. R Maidment, and L. W. Mays. *Applied Hydrology*. McGraw-Hill, 1988. [399](#), [401](#)
- [8] L. Duckstein. Multiobjective optimization in structural design: The model choice problem. In E. Atrek, R. H. Gallagher, K. M. Ragsdell, and O. C. Zienkiewicz, editors, *New Directions in Optimum Structural Design*, pages 459–481. Eds. John Wiley and Sons Inc., New York, 1984. [6](#)
- [9] G. Hartmann and A. Bárdossy. Investigation of the transferability of hydrological models and a method to improve model calibration. *Adv. Geosciences*, 5:83–87, 2005. [6](#)
- [10] Y. Hundecha and A. Bárdossy. Modeling effect of land use changes on runoff generation of a river basin through parameter regionalization of a watershed model. *J. Hydrol.*, 292:281–295, 2004. [1](#)
- [11] X. Liang, D. P. Lettenmaier, E. F. Wood, and S. J. Burges. A simple hydrologically based model of land-surface water and energy fluxes for general-circulation models. *J. Geophys. Res.-Atmos.*, 99(D7):14415–14428, 1994. [1](#)
- [12] P. Pokhrel and H. V. Gupta. On the use of spatial-regularization strategies to improve calibration of distributed watershed models. *Water Resour. Res.*, 2009. In press. [4](#)
- [13] L. Samaniego and A. Bárdossy. Robust parametric models of runoff characteristics at the mesoscale. *Journal of Hydrology*, 303(1-4):136–151, 2005. doi: 10.1016/j.jhydrol.2004.08.022. [350](#)
- [14] L. Samaniego, R. Kumar, and S. Attinger. Multiscale parameter regionalization of a grid-based hydrologic model at the mesoscale. *Water Resour. Res.*, 46, 2010. [2](#), [5](#)
- [15] B. A. Tolson and C. A. Shoemaker. Dynamically dimensioned search algorithm for computationally efficient watershed model calibration. *Water Resources Research*, 43(1):W01413, 2007. [6](#)

Index

1-main.dox, 933
2-get_started.dox, 933
3-data_preparation.dox, 933
4-visualise_out.dox, 933
5-calibration.dox, 933
6-style_guide.dox, 933
7-test_basin.dox, 933
8-protocols_for_setup_new_mHM_basin.dox, 933

absdev_dp
 mo_moment, 257
 mo_moment::absdev, 729

absdev_sp
 mo_moment, 257
 mo_moment::absdev, 729

add_inflow
 mo_mrm_routing, 398

aerodynamical_resistance
 mo_multi_param_reg, 433

all
 mo_mrm_write_fluxes_states::outputdataset, 852
 mo_write_fluxes_states::outputdataset, 856

alma_convention
 mo_common_variables, 115

anneal_dp
 mo_anneal, 75
 mo_anneal::anneal, 731

append_char_3d
 mo_append, 80
 mo_append::append, 734

append_char_m_m
 mo_append, 80
 mo_append::append, 734

append_char_v_s
 mo_append, 81
 mo_append::append, 734

append_char_v_v
 mo_append, 81
 mo_append::append, 734

append_dp_3d
 mo_append, 81
 mo_append::append, 734

append_dp_m_m
 mo_append, 81
 mo_append::append, 735

append_dp_v_s
 mo_append, 81
 mo_append::append, 735

append_dp_v_v
 mo_append, 81

approx_mon_int
 mo_append::append, 735

append_i4_3d
 mo_append, 82

append_i4_m_m
 mo_append, 82
 mo_append::append, 735

append_i4_v_s
 mo_append, 82
 mo_append::append, 735

append_i4_v_v
 mo_append, 82
 mo_append::append, 735

append_i8_3d
 mo_append, 82
 mo_append::append, 736

append_i8_m_m
 mo_append, 82
 mo_append::append, 736

append_i8_v_s
 mo_append, 82
 mo_append::append, 736

append_i8_v_v
 mo_append, 83
 mo_append::append, 736

append_lgt_3d
 mo_append, 83
 mo_append::append, 736

append_lgt_m_m
 mo_append, 83
 mo_append::append, 736

append_lgt_v_s
 mo_append, 83
 mo_append::append, 736

append_lgt_v_v
 mo_append, 83
 mo_append::append, 737

append_sp_3d
 mo_append, 83
 mo_append::append, 737

append_sp_m_m
 mo_append, 84
 mo_append::append, 737

append_sp_v_s
 mo_append, 84
 mo_append::append, 737

append_sp_v_v
 mo_append, 84
 mo_append::append, 737

mo_neutrons, 558
 approx_mon_int_eps
 mo_neutrons, 559
 approx_mon_int_steps
 mo_neutrons, 560
 arth_dp
 mo_corr, 132
 mo_corr::arth, 738
 arth_i4
 mo_corr, 132
 mo_corr::arth, 738
 arth_sp
 mo_corr, 133
 mo_corr::arth, 738
 att
 mo_ncwrite::variable, 924
 autocoeffk_1d_dp
 mo_corr, 133
 mo_corr::autocoeffk, 740
 autocoeffk_1d_sp
 mo_corr, 133
 mo_corr::autocoeffk, 740
 autocoeffk_dp
 mo_corr, 133
 mo_corr::autocoeffk, 740
 autocoeffk_sp
 mo_corr, 133
 mo_corr::autocoeffk, 740
 autocorr_1d_dp
 mo_corr, 133
 mo_corr::autocorr, 741
 autocorr_1d_sp
 mo_corr, 134
 mo_corr::autocorr, 741
 autocorr_dp
 mo_corr, 134
 mo_corr::autocorr, 741
 autocorr_sp
 mo_corr, 134
 mo_corr::autocorr, 741
 autocorrelation
 mo_mrm_signatures, 406
 average
 mo_mrm_write_fluxes_states::outputvariable, 865
 mo_write_fluxes_states::outputvariable, 860
 average_counter
 mo_mrm_write, 423
 average_dp
 mo_moment, 257
 mo_moment::average, 742
 average_sp
 mo_moment, 257
 mo_moment::average, 742
 avg
 mo_mrm_write_fluxes_states::outputvariable, 865
 mo_write_fluxes_states::outputvariable, 860
 baseflow_param
 mo_multi_param_reg, 435

basin
 mo_mrm_write_fluxes_states::outputdataset, 852
 mo_write_fluxes_states::outputdataset, 856
 basin_avg_tws_obs
 mo_global_variables, 172
 basin_avg_tws_sim
 mo_global_variables, 173
 basin_mrm
 mo_mrm_global_variables, 325
 basinid
 mo_global_variables::twsstructure, 907
 mo_mrm_global_variables::gaugingstation, 762
 before
 mo_mrm_write_fluxes_states::outputvariable, 865
 mo_write_fluxes_states::outputvariable, 860
 between
 mo_mrm_write_fluxes_states::outputvariable, 866
 mo_write_fluxes_states::outputvariable, 861
 bias_dp_1d
 mo_errormeasures, 144
 mo_errormeasures::bias, 744
 bias_dp_2d
 mo_errormeasures, 144
 mo_errormeasures::bias, 745
 bias_dp_3d
 mo_errormeasures, 144
 mo_errormeasures::bias, 745
 bias_sp_1d
 mo_errormeasures, 144
 mo_errormeasures::bias, 745
 bias_sp_2d
 mo_errormeasures, 145
 mo_errormeasures::bias, 745
 bias_sp_3d
 mo_errormeasures, 145
 mo_errormeasures::bias, 745
 bulkdens_orgmatter
 mo_mpr_constants, 263
 bulksurface_resistance
 mo_mpr_pet, 286

c1_initstatesm
 mo_mhm_constants, 245
 mo_mpr_constants, 263
 c2stu
 mo_mpr_global_variables, 278
 calculate_grid_properties
 mo_grid, 184
 caldat
 mo_julian, 194
 caldat360
 mo_julian, 196
 caldat365
 mo_julian, 196
 caldatjulian
 mo_julian, 197
 calendar
 mo_julian, 220
 calls

mo_mrm_write_fluxes_states::outputvariable, 866
mo_write_fluxes_states::outputvariable, 861
canopy_interc
 mo_canopy_interc, 87
canopy_intercept_param
 mo_multi_param_reg, 436
cce
 mo_sce, 652
cellarea
 mo_common_variables::grid, 773
cellcoor
 mo_common_variables::grid, 773
celllength
 mo_mrm_net_startup, 346
cellsize
 mo_common_variables::grid, 774
central_moment_dp
 mo_moment, 257
 mo_moment::central_moment, 746
central_moment_sp
 mo_moment, 257
 mo_moment::central_moment, 746
central_moment_var_dp
 mo_moment, 258
 mo_moment::central_moment_var, 746
central_moment_var_sp
 mo_moment, 258
 mo_moment::central_moment_var, 746
check
 mo_ncread, 446
 mo_ncwrite, 467
 mo_netcdf, 496
check_consistency_lut_map
 mo_read_wrapper, 642
check_optimization_settings
 mo_common_mhm_mrm_read_config, 97
chkcst
 mo_sce, 653
chunk_config
 mo_meteo_forcings, 231
chunk_size
 mo_meteo_forcings, 232
circum
 mo_template, 685
classified_standard_score_dp
 mo_standard_score, 673
 mo_standard_score::classified_standard_score,
 748
classified_standard_score_sp
 mo_standard_score, 673
 mo_standard_score::classified_standard_score,
 748
clay
 mo_mpr_global_variables::soiltype, 889
clock_rate
 mo_timer, 698
close
 mo_mrm_write_fluxes_states, 425
 mo_mrm_write_fluxes_states::outputdataset, 852
 mo_netcdf, 496
 mo_netcdf::ncdataset, 813
 mo_write_fluxes_states, 716
 mo_write_fluxes_states::outputdataset, 856
close_netcdf
 mo_ncwrite, 468
close_nml
 mo_nml, 567
common_check_resolution
 mo_common_mhm_mrm_read_config, 98
common_mhm_mrm_read_config
 mo_common_mhm_mrm_read_config, 98
common_read_config
 mo_common_read_config, 106
comp
 mo_sce, 653
compress
 mo_string_utils, 678
config_output
 mo_mrm_init, 334
constants_init
 mo_startup, 674
contact
 mo_common_variables, 116
contains
 mo_mrm_write_fluxes_states::outputvariable, 866
 mo_write_fluxes_states::outputvariable, 861
conventions
 mo_common_variables, 116
corr_dp
 mo_corr, 134
 mo_corr::corr, 748
corr_sp
 mo_corr, 134
 mo_corr::corr, 749
correlation_dp
 mo_moment, 258
 mo_moment::correlation, 749
correlation_sp
 mo_moment, 258
 mo_moment::correlation, 749
cosmic
 mo_neutrons, 561
cosmic_alpha
 mo_mhm_constants, 245
cosmic_bd
 mo_mhm_constants, 246
cosmic_l1
 mo_mhm_constants, 246
cosmic_l2
 mo_mhm_constants, 246
cosmic_l3
 mo_mhm_constants, 246
cosmic_l4
 mo_mhm_constants, 246
cosmic_n
 mo_mhm_constants, 246

cosmic_vwclat
 mo_mhm_constants, 246
 count
 mo_mrm_write_fluxes_states::outputdataset, 852
 mo_mrm_write_fluxes_states::outputvariable, 866
 mo_ncwrite::variable, 924
 mo_write_fluxes_states::outputdataset, 856
 mo_write_fluxes_states::outputvariable, 861
 counter
 mo_mrm_write_fluxes_states::outputdataset, 853
 mo_mrm_write_fluxes_states::outputvariable, 866
 mo_write_fluxes_states::outputdataset, 857
 mo_write_fluxes_states::outputvariable, 861
 covariance_dp
 mo_moment, 258
 mo_moment::covariance, 750
 covariance_sp
 mo_moment, 258
 mo_moment::covariance, 750
 cp0_dp
 mo_constants, 124
 cp0_sp
 mo_constants, 124
 cputime
 mo_timer, 698
 create_ncdf
 mo_ncwrite, 470
 created
 mo_mrm_write_fluxes_states::outputdataset, 853
 mo_write_fluxes_states::outputdataset, 857
 createoutputfile
 mo_mrm_write_fluxes_states, 426
 mo_write_fluxes_states, 716
 crosscoeffk_dp
 mo_corr, 134
 mo_corr::crosscoeffk, 750
 crosscoeffk_sp
 mo_corr, 135
 mo_corr::crosscoeffk, 750
 crosscorr_dp
 mo_corr, 135
 mo_corr::crosscorr, 751
 crosscorr_sp
 mo_corr, 135
 mo_corr::crosscorr, 751
 cycles1
 mo_timer, 698
 cycles2
 mo_timer, 698
 cycles_max
 mo_timer, 699
 d_ctrper
 mo_orderpack, 593
 mo_orderpack::ctrper, 752
 d_fndnth
 mo_orderpack, 593
 mo_orderpack::fndnth, 760
 d_indmed
 mo_orderpack, 593
 mo_orderpack::indmed, 779
 d_indnth
 mo_orderpack, 594
 mo_orderpack::indnth, 779
 d_inspar
 mo_orderpack, 594
 mo_orderpack::inspar, 780
 d_inssor
 mo_orderpack, 594
 mo_orderpack::inssor, 781
 d_med
 mo_orderpack, 594
 d_median
 mo_orderpack, 595
 mo_orderpack::omedian, 850
 d_mrgref
 mo_orderpack, 595
 mo_orderpack::mrgref, 807
 d_mrgrnk
 mo_orderpack, 595
 mo_orderpack::mrgrnk, 808
 d_mulcnt
 mo_orderpack, 595
 mo_orderpack::mulcnt, 810
 d_nearless
 mo_orderpack, 595
 mo_orderpack::nearless, 843
 d_rapknr
 mo_orderpack, 595
 mo_orderpack::rapknr, 877
 d_refpar
 mo_orderpack, 596
 mo_orderpack::refpar, 881
 d_refsor
 mo_orderpack, 596
 mo_orderpack::refsor, 882
 mo_orderpack::sort, 894
 d_rinpar
 mo_orderpack, 596
 mo_orderpack::rinpar, 882
 d_rnkpar
 mo_orderpack, 596
 mo_orderpack::rnkpar, 884
 d_substor
 mo_orderpack, 596
 d_uniinv
 mo_orderpack, 597
 mo_orderpack::uniinv, 908
 d_unipar
 mo_orderpack, 597
 mo_orderpack::unipar, 909
 d_unirnk
 mo_orderpack, 597
 mo_orderpack::unirnk, 909
 d_unista
 mo_orderpack, 597
 mo_orderpack::unista, 910

d_valmed
 mo_orderpack, 597
 mo_orderpack::valmed, 914

d_valnth
 mo_orderpack, 598
 mo_orderpack::valnth, 915

DEPENDENCIES.md, 933

data
 mo_mrm_write_fluxes_states::outputvariable, 866
 mo_write_fluxes_states::outputvariable, 861

dataset
 mo_netcdf::ncdataset, 821

date2dec
 mo_julian, 199

date2dec360
 mo_julian, 200

date2dec365
 mo_julian, 201

date2decjulian
 mo_julian, 203

day2mon_average_dp
 mo_temporal_aggregation, 687
 mo_temporal_aggregation::day2mon_average,
 753

day_counter
 mo_mrm_write, 424

dayhours
 mo_common_constants, 90
 mo_constants, 124

daysecs
 mo_common_constants, 90
 mo_constants, 124

db
 mo_mpr_global_variables::soiltype, 889

dbm
 mo_mpr_global_variables::soiltype, 889

dchange_dp
 mo_anneal, 76

dds
 mo_dds, 139

dds_r
 mo_common_mhm_mrm_variables, 101

dec2date
 mo_julian, 204

dec2date360
 mo_julian, 206

dec2date365
 mo_julian, 208

dec2datejulian
 mo_julian, 209

deg2rad_dp
 mo_constants, 124

deg2rad_sp
 mo_constants, 124

deltah
 mo_mrm_constants, 315

dend
 mo_common_variables::period, 876

depth
 mo_mpr_global_variables::soiltype, 889

desilets_a0
 mo_mhm_constants, 247

desilets_a1
 mo_mhm_constants, 247

desilets_a2
 mo_mhm_constants, 247

desiletsn0
 mo_neutrons, 562

dimension
 mo_netcdf::ncdimension, 841

dimid
 mo_ncwrite::dims, 754

dimids
 mo_ncwrite::variable, 924

dimtypes
 mo_ncwrite::variable, 924

dirabsvappressure
 mo_global_variables, 173

dircommonfiles
 mo_common_variables, 116

dirconfigout
 mo_common_variables, 116

direvapotranspiration
 mo_global_variables, 173

dirgauges
 mo_mrm_global_variables, 325

dirgridded_lai
 mo_mpr_global_variables, 278

dirlcover
 mo_common_variables, 116

dirmaxtemperature
 mo_global_variables, 173

dirmintemperature
 mo_global_variables, 173

dirmorpho
 mo_common_variables, 116

dirnetradiation
 mo_global_variables, 173

dirneutrons
 mo_global_variables, 173

dirout
 mo_common_variables, 116

dirprecipitation
 mo_global_variables, 174

dirreferenceet
 mo_global_variables, 174

dirrestartin
 mo_common_mhm_mrm_variables, 101

dirrestartout
 mo_common_variables, 117

dirsoil_moisture
 mo_global_variables, 174

dirtemperature
 mo_global_variables, 174

dirtotalrunoff
 mo_mrm_global_variables, 325

dirwindspeed
 mo_global_variables, 174

divide_string
 mo_string_utils, 679

dnc
 mo_ncwrite, 491

dp
 mo_kind, 222

dp2str
 mo_string_utils, 680
 mo_string_utils::num2str, 848

dpc
 mo_kind, 222

dstart
 mo_common_variables::period, 876

duffiedelta1
 mo_mhm_constants, 247

duffiedelta2
 mo_mhm_constants, 247

duffiedr
 mo_mhm_constants, 247

dummy_mpr, 75

dummy_mrm, 75

dump_ncdf_1d_dp
 mo_ncwrite, 471
 mo_ncwrite::dump_ncdf, 754

dump_ncdf_1d_i4
 mo_ncwrite, 471
 mo_ncwrite::dump_ncdf, 755

dump_ncdf_1d_sp
 mo_ncwrite, 472
 mo_ncwrite::dump_ncdf, 755

dump_ncdf_2d_dp
 mo_ncwrite, 472
 mo_ncwrite::dump_ncdf, 755

dump_ncdf_2d_i4
 mo_ncwrite, 473
 mo_ncwrite::dump_ncdf, 755

dump_ncdf_2d_sp
 mo_ncwrite, 473
 mo_ncwrite::dump_ncdf, 756

dump_ncdf_3d_dp
 mo_ncwrite, 474
 mo_ncwrite::dump_ncdf, 756

dump_ncdf_3d_i4
 mo_ncwrite, 474
 mo_ncwrite::dump_ncdf, 756

dump_ncdf_3d_sp
 mo_ncwrite, 475
 mo_ncwrite::dump_ncdf, 756

dump_ncdf_4d_dp
 mo_ncwrite, 475
 mo_ncwrite::dump_ncdf, 756

dump_ncdf_4d_i4
 mo_ncwrite, 476
 mo_ncwrite::dump_ncdf, 757

dump_ncdf_4d_sp
 mo_ncwrite, 476

mo_ncwrite::dump_ncdf, 757

mo_ncwrite::dump_ncdf_5d_dp
 mo_ncwrite, 477

mo_ncwrite::dump_ncdf_5d_i4
 mo_ncwrite, 477

mo_ncwrite::dump_ncdf_5d_sp
 mo_ncwrite, 478

mo_ncwrite::dump_ncdf, 757

mo_ncwrite::dump_ncdf_5d_sp
 mo_ncwrite, 478

mo_ncwrite::dump_ncdf, 757

eps_dp
 mo_common_constants, 90

eps_sp
 mo_common_constants, 90

equal_dp
 mo_utils, 707
 mo_utils::eq, 758
 mo_utils::equal, 759

equal_sp
 mo_utils, 707
 mo_utils::eq, 758
 mo_utils::equal, 759

equalncdimensions
 mo_ncdf, 497

equalstrings
 mo_string_utils, 680

euler
 mo_constants, 125

euler_d
 mo_constants, 125

eval_interface
 mo_optimization_utils::eval_interface, 760

evalper
 mo_common_mhm_mrm_variables, 102

evap_coeff
 mo_global_variables, 174

extract_basin_avg_tws
 mo_objective_function, 573

extract_runoff
 mo_mrm_objective_function_runoff, 360

extraterr_rad_approx
 mo_pet, 611

fday_pet
 mo_global_variables, 174

fday_prec
 mo_global_variables, 175

fday_temp
 mo_global_variables, 175

feddes_et_reduction
 mo_soil_moisture, 665

field_cap
 mo_mpr_soilmoist, 303

field_cap_c1
 mo_mpr_constants, 263

field_cap_c2
 mo_mpr_constants, 263

file

mo_ncdf::ncdataset, 821
file_aspect
 mo_mpr_file, 272
file_config
 mo_common_file, 93
 mo_mrm_file, 320
file_daily_discharge
 mo_mrm_file, 320
file_defoutput
 mo_file, 167
 mo_mrm_file, 320
file_dem
 mo_common_file, 93
file_facc
 mo_mrm_file, 320
file_fdir
 mo_mrm_file, 320
file_gaugeloc
 mo_mrm_file, 320
file_geolut
 mo_mpr_file, 272
file_hydrogeoclass
 mo_mpr_file, 272
file_laiclass
 mo_mpr_file, 272
file_lailut
 mo_mpr_file, 273
file_main
 mo_file, 168
 mo_mpr_file, 273
 mo_mrm_file, 320
file_meteo_binary_end
 mo_mpr_file, 273
file_meteo_header
 mo_mpr_file, 273
file_mrm_output
 mo_mrm_file, 321
file_namelist_mhm
 mo_file, 168
file_namelist_mhm_param
 mo_file, 168
file_namelist_mpr
 mo_mpr_file, 273
file_namelist_mpr_param
 mo_mpr_file, 273
file_namelist_mrm
 mo_mrm_file, 321
file_namelist_param_mrm
 mo_mrm_file, 321
file_opti
 mo_common_mhm_mrm_file, 95
file_opti_nml
 mo_common_mhm_mrm_file, 96
file_slope
 mo_mpr_file, 273
file_soil_database
 mo_mpr_file, 274
file_soil_database_1
 mo_mpr_file, 274
file_soilclass
 mo_mpr_file, 274
filelatlon
 mo_common_variables, 117
filename
 mo_ncdf::ncdataset, 821
filenametotalrunoff
 mo_mrm_global_variables, 325
filetw5
 mo_global_variables, 175
finish
 mo_finish, 169
flowdurationcurve
 mo_mrm_signatures, 406
fluxesunit
 mo_write_fluxes_states, 717
fname
 mo_global_variables::twsstructure, 907
 mo_mrm_global_variables::gaugingstation, 762
 mo_ncdf::ncdataset, 821
fnight_pet
 mo_global_variables, 175
fnight_prec
 mo_global_variables, 175
fnight_temp
 mo_global_variables, 175
four1_dp
 mo_corr, 135
 mo_corr::four1, 761
four1_sp
 mo_corr, 135
 mo_corr::four1, 761
fourrow_dp
 mo_corr, 135
 mo_corr::fourrow, 761
fourrow_sp
 mo_corr, 136
 mo_corr::fourrow, 761
fracsealed_cityarea
 mo_mpr_global_variables, 278
g0_b
 mo_ncwrite::variable, 925
g0_d
 mo_ncwrite::variable, 925
g0_f
 mo_ncwrite::variable, 925
g0_i
 mo_ncwrite::variable, 925
g1_b
 mo_ncwrite::variable, 925
g1_d
 mo_ncwrite::variable, 925
g1_f
 mo_ncwrite::variable, 925
g1_i
 mo_ncwrite::variable, 925
g2_b

mo_ncwrite::variable, 925
 g2_d
 mo_ncwrite::variable, 926
 g2_f
 mo_ncwrite::variable, 926
 g2_i
 mo_ncwrite::variable, 926
 g3_b
 mo_ncwrite::variable, 926
 g3_d
 mo_ncwrite::variable, 926
 g3_f
 mo_ncwrite::variable, 926
 g3_i
 mo_ncwrite::variable, 926
 g4_b
 mo_ncwrite::variable, 926
 g4_d
 mo_ncwrite::variable, 926
 g4_f
 mo_ncwrite::variable, 927
 g4_i
 mo_ncwrite::variable, 927
 gatt
 mo_ncwrite, 491
 gauge
 mo_mrm_global_variables, 325
 gaugeid
 mo_mrm_global_variables::gaugingstation, 762
 gaugeidlist
 mo_mrm_global_variables::basininfo_mrm, 743
 gaugeindexlist
 mo_mrm_global_variables::basininfo_mrm, 743
 gaugenodelist
 mo_mrm_global_variables::basininfo_mrm, 743
 generate_neighborhood_weight_dp
 mo_anneal, 77
 mo_anneal::generate_neighborhood_weight, 763
 generate_soil_database
 mo_soil_database, 663
 generatenewparameterset_dp
 mo_mcmc, 225
 genuchten
 mo_mpr_soilmoist, 304
 geocoordinates
 mo_grid, 186
 geounitkar
 mo_mpr_global_variables, 278
 geounitlist
 mo_mpr_global_variables, 278
 get_distance_two_lat_lon_points
 mo_mrm_net_startup, 347
 get_info
 mo_ncread, 447
 get_ncdim
 mo_ncread, 449
 get_ncdimatt
 mo_ncread, 451
 get_ncvar_0d_dp
 mo_ncread, 452
 mo_ncread::get_ncvar, 764
 get_ncvar_0d_i1
 mo_ncread, 452
 mo_ncread::get_ncvar, 764
 get_ncvar_0d_i4
 mo_ncread, 453
 mo_ncread::get_ncvar, 765
 get_ncvar_0d_sp
 mo_ncread, 453
 mo_ncread::get_ncvar, 765
 get_ncvar_1d_dp
 mo_ncread, 454
 mo_ncread::get_ncvar, 765
 get_ncvar_1d_i1
 mo_ncread, 454
 mo_ncread::get_ncvar, 765
 get_ncvar_1d_i4
 mo_ncread, 455
 mo_ncread::get_ncvar, 765
 get_ncvar_1d_sp
 mo_ncread, 455
 mo_ncread::get_ncvar, 766
 get_ncvar_2d_dp
 mo_ncread, 456
 mo_ncread::get_ncvar, 766
 get_ncvar_2d_i1
 mo_ncread, 456
 mo_ncread::get_ncvar, 766
 get_ncvar_2d_i4
 mo_ncread, 457
 mo_ncread::get_ncvar, 766
 get_ncvar_2d_sp
 mo_ncread, 457
 mo_ncread::get_ncvar, 766
 get_ncvar_3d_dp
 mo_ncread, 458
 mo_ncread::get_ncvar, 767
 get_ncvar_3d_i1
 mo_ncread, 458
 mo_ncread::get_ncvar, 767
 get_ncvar_3d_i4
 mo_ncread, 459
 mo_ncread::get_ncvar, 767
 get_ncvar_3d_sp
 mo_ncread, 459
 mo_ncread::get_ncvar, 767
 get_ncvar_4d_dp
 mo_ncread, 460
 mo_ncread::get_ncvar, 768
 get_ncvar_4d_i1
 mo_ncread, 460
 mo_ncread::get_ncvar, 768
 get_ncvar_4d_i4
 mo_ncread, 461
 mo_ncread::get_ncvar, 768
 get_ncvar_4d_sp

mo_ncread, 461
mo_ncread::get_ncvar, 768
get_ncvar_5d_dp
mo_ncread, 462
mo_ncread::get_ncvar, 768
get_ncvar_5d_i1
mo_ncread, 462
mo_ncread::get_ncvar, 769
get_ncvar_5d_i4
mo_ncread, 463
mo_ncread::get_ncvar, 769
get_ncvar_5d_sp
mo_ncread, 463
mo_ncread::get_ncvar, 769
get_ncvaratt
mo_ncread, 464
get_time_vector_and_select
mo_read_forcing_nc, 622
get_timeseed_i4_0d
mo_xor4096, 725
mo_xor4096::get_timeseed, 769
get_timeseed_i4_1d
mo_xor4096, 725
mo_xor4096::get_timeseed, 770
get_timeseed_i8_0d
mo_xor4096, 725
mo_xor4096::get_timeseed, 770
get_timeseed_i8_1d
mo_xor4096, 725
mo_xor4096::get_timeseed, 770
getattribute
mo_ncdf::ncdataset, 813
mo_ncdf::ncdimension, 826
getdata
mo_ncdf::ncdimension, 827
getdata1df32
mo_ncdf, 497
mo_ncdf::ncdimension, 827
getdata1df64
mo_ncdf, 497
mo_ncdf::ncdimension, 827
getdata1di16
mo_ncdf, 498
mo_ncdf::ncdimension, 827
getdata1di32
mo_ncdf, 498
mo_ncdf::ncdimension, 827
getdata1di64
mo_ncdf, 499
mo_ncdf::ncdimension, 827
getdata1di8
mo_ncdf, 499
mo_ncdf::ncdimension, 828
getdata2df32
mo_ncdf, 500
mo_ncdf::ncdimension, 828
getdata2df64
mo_ncdf, 500

mo_ncdf::ncdimension, 828
getdata2di16
mo_ncdf, 501
mo_ncdf::ncdimension, 828
getdata2di32
mo_ncdf, 501
mo_ncdf::ncdimension, 828
getdata2di64
mo_ncdf, 502
mo_ncdf::ncdimension, 828
getdata2di8
mo_ncdf, 502
mo_ncdf::ncdimension, 828
getdata3df32
mo_ncdf, 503
mo_ncdf::ncdimension, 828
getdata3df64
mo_ncdf, 503
mo_ncdf::ncdimension, 828
getdata3di16
mo_ncdf, 504
mo_ncdf::ncdimension, 829
getdata3di32
mo_ncdf, 504
mo_ncdf::ncdimension, 829
getdata3di64
mo_ncdf, 505
mo_ncdf::ncdimension, 829
getdata3di8
mo_ncdf, 505
mo_ncdf::ncdimension, 829
getdata4df32
mo_ncdf, 506
mo_ncdf::ncdimension, 829
getdata4df64
mo_ncdf, 506
mo_ncdf::ncdimension, 829
getdata4di16
mo_ncdf, 507
mo_ncdf::ncdimension, 829
getdata4di32
mo_ncdf, 507
mo_ncdf::ncdimension, 829
getdata4di64
mo_ncdf, 508
mo_ncdf::ncdimension, 829
getdata4di8
mo_ncdf, 508
mo_ncdf::ncdimension, 830
getdata5df32
mo_ncdf, 509
mo_ncdf::ncdimension, 830
getdata5df64
mo_ncdf, 509
mo_ncdf::ncdimension, 830
getdata5di16
mo_ncdf, 510
mo_ncdf::ncdimension, 830

getdata5di32
 mo_netcdf, 510
 mo_netcdf::ncdimension, 830

getdata5di64
 mo_netcdf, 511
 mo_netcdf::ncdimension, 830

getdata5di8
 mo_netcdf, 511
 mo_netcdf::ncdimension, 830

getdatascalarf32
 mo_netcdf, 512
 mo_netcdf::ncdimension, 830

getdatascalarf64
 mo_netcdf, 512
 mo_netcdf::ncdimension, 830

getdatascalari16
 mo_netcdf, 513
 mo_netcdf::ncdimension, 831

getdatascalari32
 mo_netcdf, 513
 mo_netcdf::ncdimension, 831

getdatascalari64
 mo_netcdf, 514
 mo_netcdf::ncdimension, 831

getdatascalari8
 mo_netcdf, 514
 mo_netcdf::ncdimension, 831

getdimension
 mo_netcdf::ncdataset, 814

getdimensionbyid
 mo_netcdf, 515
 mo_netcdf::ncdataset, 814

getdimensionbyname
 mo_netcdf, 515
 mo_netcdf::ncdataset, 814

getdimensionlength
 mo_netcdf, 515

getdimensionname
 mo_netcdf, 516

getdimensions
 mo_netcdf::ncdimension, 831

getdtype
 mo_netcdf::ncdimension, 831

getdtypefrominteger
 mo_netcdf, 516

getdtypefromstring
 mo_netcdf, 517

getfillvalue
 mo_netcdf::ncdimension, 831

getglobalattributechar
 mo_netcdf, 517
 mo_netcdf::ncdataset, 815

getglobalattributef32
 mo_netcdf, 517
 mo_netcdf::ncdataset, 815

getglobalattributef64
 mo_netcdf, 518
 mo_netcdf::ncdataset, 815

getglobalattributei16
 mo_netcdf, 518
 mo_netcdf::ncdataset, 815

getglobalattributei32
 mo_netcdf, 519
 mo_netcdf::ncdataset, 815

getglobalattributei64
 mo_netcdf, 519
 mo_netcdf::ncdataset, 815

getglobalattributei8
 mo_netcdf, 520
 mo_netcdf::ncdataset, 815

getname
 mo_netcdf::ncdimension, 832

getnondimensions
 mo_netcdf, 520
 mo_netcdf::ncdimension, 832

getnovariables
 mo_netcdf, 520
 mo_netcdf::ncdataset, 815

getpnt
 mo_sce, 654

getreaddatashape
 mo_netcdf, 521

getshape
 mo_netcdf::ncdimension, 832

gettemperature_dp
 mo_anneal, 77
 mo_anneal::gettemperature, 771

getunlimiteddimension
 mo_netcdf, 522
 mo_netcdf::ncdataset, 815

getvariable
 mo_netcdf::ncdataset, 816

getvariableattributechar
 mo_netcdf, 523
 mo_netcdf::ncdimension, 832

getvariableattributef32
 mo_netcdf, 523
 mo_netcdf::ncdimension, 832

getvariableattributef64
 mo_netcdf, 524
 mo_netcdf::ncdimension, 832

getvariableattributei16
 mo_netcdf, 524
 mo_netcdf::ncdimension, 832

getvariableattributei32
 mo_netcdf, 524
 mo_netcdf::ncdimension, 832

getvariableattributei64
 mo_netcdf, 525
 mo_netcdf::ncdimension, 833

getvariableattributei8
 mo_netcdf, 525
 mo_netcdf::ncdimension, 833

getvariablebyname
 mo_netcdf, 526
 mo_netcdf::ncdataset, 816

getvariableldimensions
 mo_netcdf, 526

getvariableldtype
 mo_netcdf, 526

getvariablefillvaluef32
 mo_netcdf, 527
 mo_netcdf::ncdimension, 833

getvariablefillvaluef64
 mo_netcdf, 527
 mo_netcdf::ncdimension, 833

getvariablefillvaluei16
 mo_netcdf, 527
 mo_netcdf::ncdimension, 833

getvariablefillvaluei32
 mo_netcdf, 527
 mo_netcdf::ncdimension, 833

getvariablefillvaluei64
 mo_netcdf, 527
 mo_netcdf::ncdimension, 833

getvariablefillvaluei8
 mo_netcdf, 527
 mo_netcdf::ncdimension, 833

getvariableids
 mo_netcdf, 528
 mo_netcdf::ncdataset, 816

getvariablelename
 mo_netcdf, 528

getvariables
 mo_netcdf, 528
 mo_netcdf::ncdataset, 817

getvariableshape
 mo_netcdf, 528

given_ts
 mo_mrm_constants, 315

global_parameters
 mo_common_variables, 117

global_parameters_name
 mo_common_variables, 117

gravity_dp
 mo_constants, 125

gravity_sp
 mo_constants, 125

greaterequal_dp
 mo_utils, 707
 mo_utils::ge, 763
 mo_utils::greaterequal, 772

greaterequal_sp
 mo_utils, 707
 mo_utils::ge, 763
 mo_utils::greaterequal, 772

h2odens
 mo_mhm_constants, 247

harsamconst
 mo_mhm_constants, 247

hasattribute
 mo_netcdf, 529
 mo_netcdf::ncdimension, 833

hasdimension

 mo_netcdf, 529
 mo_netcdf::ncdataset, 817

hasvariable
 mo_netcdf, 529
 mo_netcdf::ncdataset, 817

high_res_grid
 mo_common_variables::gridremapper, 776

history
 mo_common_variables, 117

horizondepth_mhm
 mo_mpr_global_variables, 278

hour2day_average_dp
 mo_temporal_aggregation, 688
 mo_temporal_aggregation::hour2day_average, 778

hoursecs
 mo_common_constants, 90

hydro_cond
 mo_mpr_soilmoist, 305

i1
 mo_kind, 222

i2
 mo_kind, 222

i4
 mo_kind, 222

i42str
 mo_string_utils, 680
 mo_string_utils::num2str, 848

i4array2str
 mo_string_utils, 681
 mo_string_utils::numarray2str, 849

i8
 mo_kind, 223

i82str
 mo_string_utils, 681
 mo_string_utils::num2str, 848

i_ctrper
 mo_orderpack, 598
 mo_orderpack::ctrper, 752

i_fndnth
 mo_orderpack, 598
 mo_orderpack::fndnth, 760

i_indmed
 mo_orderpack, 598
 mo_orderpack::indmed, 779

i_indnth
 mo_orderpack, 598
 mo_orderpack::indnth, 779

i_inspar
 mo_orderpack, 599
 mo_orderpack::inspar, 780

i_inssor
 mo_orderpack, 599
 mo_orderpack::inssor, 781

i_med
 mo_orderpack, 599

i_median
 mo_orderpack, 599

mo_orderpack::omedian, 850
 i_mrgref
 mo_orderpack, 600
 mo_orderpack::mrgref, 807
 i_mrgrnk
 mo_orderpack, 600
 mo_orderpack::mrgrnk, 808
 i_mulcnt
 mo_orderpack, 600
 mo_orderpack::mulcnt, 810
 i_nearless
 mo_orderpack, 600
 mo_orderpack::nearless, 843
 i_rapknr
 mo_orderpack, 600
 mo_orderpack::rapknr, 878
 i_refpar
 mo_orderpack, 600
 mo_orderpack::refpar, 881
 i_refsor
 mo_orderpack, 600
 mo_orderpack::refsor, 882
 mo_orderpack::sort, 894
 i_rinpar
 mo_orderpack, 601
 mo_orderpack::rinpar, 883
 i_rnkpar
 mo_orderpack, 601
 mo_orderpack::rnkpar, 885
 i_subsov
 mo_orderpack, 601
 i_uniinv
 mo_orderpack, 602
 mo_orderpack::uniinv, 908
 i_unipar
 mo_orderpack, 602
 mo_orderpack::unipar, 909
 i_unirnk
 mo_orderpack, 602
 mo_orderpack::unirnk, 910
 i_unista
 mo_orderpack, 602
 mo_orderpack::unista, 910
 i_valmed
 mo_orderpack, 602
 mo_orderpack::valmed, 914
 i_valnth
 mo_orderpack, 602
 mo_orderpack::valnth, 915
 ibasin
 mo_mrm_write_fluxes_states::outputdataset, 853
 mo_write_fluxes_states::outputdataset, 857
 id
 mo_common_variables::grid, 774
 mo_mpr_global_variables::soiltype, 889
 mo_mrm_write_fluxes_states::outputdataset, 853
 mo_ncdf::ncdataset, 821
 mo_ncdf::ncdimension, 841
 mo_write_fluxes_states::outputdataset, 857
 idont
 mo_orderpack, 608
 iend
 mo_common_variables::grid, 774
 iflag_coordinate_sys
 mo_common_variables, 117
 iflag_soildb
 mo_mpr_global_variables, 279
 in_bound
 mo_common_functions, 94
 inflowgauge
 mo_mrm_global_variables, 325
 inflowgaugeheadwater
 mo_mrm_global_variables::basininfo_mrm, 743
 inflowgaugeidlist
 mo_mrm_global_variables::basininfo_mrm, 743
 inflowgaugeindexlist
 mo_mrm_global_variables::basininfo_mrm, 743
 inflowgaugenodelist
 mo_mrm_global_variables::basininfo_mrm, 743
 init_eff_params
 mo_mpr_startup, 310
 init_lowres_level
 mo_grid, 186
 initncdataset
 mo_ncdf, 529
 mo_ncdf::ncdataset, 818
 initncdimension
 mo_ncdf, 529
 initncvariable
 mo_ncdf, 530
 mo_ncdf::ncdimension, 834
 inputformat_gridded_lai
 mo_mpr_global_variables, 279
 inputformat_meteo_forcings
 mo_global_variables, 175
 intgrandfast
 mo_neutrons, 563
 iper_thres_runoff
 mo_multi_param_reg, 437
 irow
 mo_kind::sprs2_dp, 899
 mo_kind::sprs2_sp, 901
 is_finite_dp
 mo_utils, 708
 mo_utils::is_finite, 782
 is_finite_sp
 mo_utils, 708
 mo_utils::is_finite, 782
 is_nan_dp
 mo_utils, 708
 mo_utils::is_nan, 782
 is_nan_sp
 mo_utils, 708
 mo_utils::is_nan, 782
 is_normal_dp
 mo_utils, 708

mo_utils::is_normal, 783
is_normal_sp
 mo_utils, 708
 mo_utils::is_normal, 783
is_present
 mo_mpr_global_variables::soiltype, 889
is_read
 mo_meteo_forcings, 233
is_start
 mo_mrm_global_variables, 326
isdatasetunlimited
 mo_netcdf, 530
istart
 mo_common_variables::grid, 774
isunlimited
 mo_netcdf::ncdataset, 818
 mo_netcdf::ncdimension, 834
isunlimitedddimension
 mo_netcdf, 530
isunlimitedvariable
 mo_netcdf, 530
itest
 mo_template, 686
jarvis_et_reduction
 mo_soil_moisture, 666
jcol
 mo_kind::sprs2_dp, 899
 mo_kind::sprs2_sp, 901
julday
 mo Julian, 211
julday360
 mo Julian, 212
julday365
 mo Julian, 213
juldayjulian
 mo Julian, 214
julend
 mo_common_variables::period, 876
julstart
 mo_common_variables::period, 876
karman
 mo_mpr_constants, 263
karstic_layer
 mo_multi_param_reg, 438
kge_dp_1d
 mo_errormeasures, 145
 mo_errormeasures::kge, 784
kge_dp_2d
 mo_errormeasures, 145
 mo_errormeasures::kge, 784
kge_dp_3d
 mo_errormeasures, 145
 mo_errormeasures::kge, 785
kge_sp_1d
 mo_errormeasures, 145
 mo_errormeasures::kge, 785
kge_sp_2d
 mo_errormeasures, 146
 mo_errormeasures::kge, 785
kge_sp_3d
 mo_errormeasures, 146
 mo_errormeasures::kge, 785
kgenocorr_dp_1d
 mo_errormeasures, 146
 mo_errormeasures::kgenocorr, 786
kgenocorr_dp_2d
 mo_errormeasures, 146
 mo_errormeasures::kgenocorr, 786
kgenocorr_dp_3d
 mo_errormeasures, 146
 mo_errormeasures::kgenocorr, 787
kgenocorr_sp_1d
 mo_errormeasures, 146
 mo_errormeasures::kgenocorr, 787
kgenocorr_sp_2d
 mo_errormeasures, 147
 mo_errormeasures::kgenocorr, 787
kgenocorr_sp_3d
 mo_errormeasures, 147
 mo_errormeasures::kgenocorr, 787
ks
 mo_mpr_global_variables::soiltype, 890
ks_c
 mo_mpr_constants, 263
kurtosis_dp
 mo_moment, 259
 mo_moment::kurtosis, 788
kurtosis_sp
 mo_moment, 259
 mo_moment::kurtosis, 788
l0_asp
 mo_mpr_global_variables, 279
l0_basin
 mo_common_variables, 118
l0_check_input
 mo_mpr_startup, 310
l0_check_input_routing
 mo_mrm_init, 335
l0_coloutlet
 mo_mrm_global_variables::basininfo_mrm, 744
l0_dracell
 mo_mrm_global_variables, 326
l0_drasc
 mo_mrm_global_variables, 326
l0_elev
 mo_common_variables, 118
l0_facc
 mo_mrm_global_variables, 326
l0_fdir
 mo_mrm_global_variables, 326
l0_floodplain
 mo_mrm_global_variables, 326
l0_fractionalcover_in_lx
 mo_upscaling_operators, 700
l0_gaugeloc

mo_mrm_global_variables, 326
 l0_geounit
 mo_mpr_global_variables, 279
 l0_grid_setup
 mo_grid, 187
 l0_gridded_lai
 mo_mpr_global_variables, 279
 l0_inflowgaugeloc
 mo_mrm_global_variables, 327
 l0_l11_remap
 mo_mrm_global_variables, 327
 l0_l1_remap
 mo_common_variables, 118
 l0_lcover
 mo_common_variables, 118
 l0_noutlet
 mo_mrm_global_variables::basininfo_mrm, 744
 l0_rowoutlet
 mo_mrm_global_variables::basininfo_mrm, 744
 l0_slope
 mo_mpr_global_variables, 279
 l0_slope_emp
 mo_mpr_global_variables, 280
 l0_soilid
 mo_mpr_global_variables, 280
 l0_streamnet
 mo_mrm_global_variables, 327
 l0_variable_init
 mo_mpr_startup, 311
 l11_afloodplain
 mo_mrm_global_variables, 327
 l11_c1
 mo_mrm_global_variables, 327
 l11_c2
 mo_mrm_global_variables, 327
 l11_colout
 mo_mrm_global_variables, 328
 l11_fcol
 mo_mrm_global_variables, 328
 l11_fdir
 mo_mrm_global_variables, 328
 l11_flow_direction
 mo_mrm_net_startup, 348
 l11_fraction_sealed_floodplain
 mo_mrm_net_startup, 350
 l11_fromn
 mo_mrm_global_variables, 328
 l11_frow
 mo_mrm_global_variables, 328
 l11_k
 mo_mrm_global_variables, 328
 l11_l1_id
 mo_mrm_global_variables, 328
 l11_l1_mapping
 mo_mrm_net_startup, 351
 l11_label
 mo_mrm_global_variables, 329
 l11_length
 mo_mrm_global_variables, 329
 l11_link_location
 mo_mrm_net_startup, 352
 l11_netperm
 mo_mrm_global_variables, 329
 l11_nlinkfracfpimp
 mo_mrm_global_variables, 329
 l11_noutlets
 mo_mrm_global_variables, 329
 l11_qmod
 mo_mrm_global_variables, 329
 l11_qout
 mo_mrm_global_variables, 330
 l11_qtin
 mo_mrm_global_variables, 330
 l11_qtr
 mo_mrm_global_variables, 330
 l11_rorder
 mo_mrm_global_variables, 330
 l11_routing
 mo_mrm_routing, 399
 l11_routing_order
 mo_mrm_net_startup, 353
 l11_rowout
 mo_mrm_global_variables, 330
 l11_runoff_acc
 mo_mrm_routing, 401
 l11_set_drain_outlet_gauges
 mo_mrm_net_startup, 354
 l11_set_network_topology
 mo_mrm_net_startup, 355
 l11_sink
 mo_mrm_global_variables, 330
 l11_slope
 mo_mrm_global_variables, 331
 l11_stream_features
 mo_mrm_net_startup, 356
 l11_tcol
 mo_mrm_global_variables, 331
 l11_ton
 mo_mrm_global_variables, 331
 l11_trow
 mo_mrm_global_variables, 331
 l11_tsroutr
 mo_mrm_global_variables, 331
 l11_xi
 mo_mrm_global_variables, 331
 l1_absvappress
 mo_global_variables, 176
 l1_aeroresist
 mo_mpr_global_variables, 280
 l1_aetcanyopy
 mo_global_variables, 176
 l1_aetsealed
 mo_global_variables, 176
 l1_aetsoil
 mo_global_variables, 176
 l1_alpha

mo_mpr_global_variables, 280
l1_baseflow
 mo_global_variables, 176
l1_degday
 mo_mpr_global_variables, 280
l1_degdayinc
 mo_mpr_global_variables, 280
l1_degdaymax
 mo_mpr_global_variables, 280
l1_degdaynopre
 mo_mpr_global_variables, 281
l1_et
 mo_global_variables, 176
l1_et_mask
 mo_global_variables, 176
l1_fasp
 mo_mpr_global_variables, 281
l1_fastrunoff
 mo_global_variables, 177
l1_froots
 mo_mpr_global_variables, 281
l1_fsealed
 mo_mpr_global_variables, 281
l1_harsamcoeff
 mo_mpr_global_variables, 281
l1_infilsoil
 mo_global_variables, 177
l1_inter
 mo_global_variables, 177
l1_jarvis_thresh_c1
 mo_mpr_global_variables, 281
l1_karstloss
 mo_mpr_global_variables, 282
l1_kbaseflow
 mo_mpr_global_variables, 282
l1_kfastflow
 mo_mpr_global_variables, 282
l1_kperco
 mo_mpr_global_variables, 282
l1_kslowflow
 mo_mpr_global_variables, 282
l1_l11_id
 mo_mrm_global_variables, 332
l1_l11_remap
 mo_mrm_global_variables, 332
l1_maxinter
 mo_mpr_global_variables, 282
l1_melt
 mo_global_variables, 177
l1_netrad
 mo_global_variables, 177
l1_neutrons
 mo_global_variables, 177
l1_neutronsdata
 mo_global_variables, 177
l1_neutronsdata_mask
 mo_global_variables, 178
l1_percol
 mo_global_variables, 178
l1_pet
 mo_global_variables, 178
l1_pet_calc
 mo_global_variables, 178
l1_pet_weights
 mo_global_variables, 178
l1_petlaicorfactor
 mo_mpr_global_variables, 282
l1_pre
 mo_global_variables, 178
l1_pre_weights
 mo_global_variables, 178
l1_preeffect
 mo_global_variables, 179
l1_prietaryalpha
 mo_mpr_global_variables, 283
l1_rain
 mo_global_variables, 179
l1_runoffseal
 mo_global_variables, 179
l1_satstw
 mo_global_variables, 179
l1_sealedthresh
 mo_mpr_global_variables, 283
l1_sealstw
 mo_global_variables, 179
l1_slowrunoff
 mo_global_variables, 179
l1_sm
 mo_global_variables, 179
l1_sm_mask
 mo_global_variables, 180
l1_snow
 mo_global_variables, 180
l1_snowpack
 mo_global_variables, 180
l1_soilmoist
 mo_global_variables, 180
l1_soilmoistexp
 mo_mpr_global_variables, 283
l1_soilmoistfc
 mo_mpr_global_variables, 283
l1_soilmoistsat
 mo_mpr_global_variables, 283
l1_surfresist
 mo_mpr_global_variables, 283
l1_temp
 mo_global_variables, 180
l1_temp_weights
 mo_global_variables, 180
l1_tempthresh
 mo_mpr_global_variables, 283
l1_throughfall
 mo_global_variables, 181
l1_tmax
 mo_global_variables, 181
l1_tmin

mo_global_variables, 181
 l1_total_runoff
 mo_global_variables, 181
 mo_runoff, 649
 l1_total_runoff_in
 mo_mrm_global_variables, 332
 l1_unsatstw
 mo_global_variables, 181
 l1_unsatthresh
 mo_mpr_global_variables, 284
 l1_wiltingpoint
 mo_mpr_global_variables, 284
 l1_windspeed
 mo_global_variables, 181
 l2_variable_init
 mo_startup, 675
 lai_factor_surfresi
 mo_mpr_constants, 263
 lai_offset_surfresi
 mo_mpr_constants, 264
 laiut
 mo_mpr_global_variables, 284
 laiper
 mo_mpr_global_variables, 284
 laiunitlist
 mo_mpr_global_variables, 284
 lc_year_end
 mo_common_variables, 118
 lc_year_start
 mo_common_variables, 118
 lcfilename
 mo_common_variables, 119
 lcyearid
 mo_common_mhm_mrm_variables, 102
 ld
 mo_mpr_global_variables::soiltype, 890
 left_bound
 mo_common_variables::gridremapper, 777
 len
 mo_kind::sprs2_dp, 900
 mo_kind::sprs2_sp, 901
 mo_ncwrite::dims, 754
 length_error
 mo_nml, 571
 lesserequal_dp
 mo_utils, 708
 mo_utils::le, 788
 mo_utils::lesserequal, 789
 lesserequal_sp
 mo_utils, 708
 mo_utils::le, 788
 mo_utils::lesserequal, 789
 level0
 mo_common_variables, 119
 level1
 mo_common_variables, 119
 level11
 mo_mrm_global_variables, 332
 level2
 mo_global_variables, 181
 lgt
 mo_kind, 223
 limb_densities
 mo_mrm_signatures, 408
 linfit_dp
 mo_linfit, 224
 mo_linfit::linfit, 790
 linfit_sp
 mo_linfit, 224
 mo_linfit::linfit, 790
 lnse_dp_1d
 mo_errormeasures, 147
 mo_errormeasures::lnse, 791
 lnse_dp_2d
 mo_errormeasures, 147
 mo_errormeasures::lnse, 791
 lnse_dp_3d
 mo_errormeasures, 147
 mo_errormeasures::lnse, 791
 lnse_sp_1d
 mo_errormeasures, 147
 mo_errormeasures::lnse, 791
 lnse_sp_2d
 mo_errormeasures, 147
 mo_errormeasures::lnse, 791
 lnse_sp_3d
 mo_errormeasures, 148
 mo_errormeasures::lnse, 791
 locate_0d_dp
 mo_utils, 709
 mo_utils::locate, 792
 locate_0d_sp
 mo_utils, 709
 mo_utils::locate, 792
 locate_1d_dp
 mo_utils, 709
 mo_utils::locate, 793
 locate_1d_sp
 mo_utils, 709
 mo_utils::locate, 793
 log2str
 mo_string_utils, 681
 mo_string_utils::num2str, 848
 loglikelihood_evin2013_2
 mo_mrm_objective_function_runoff, 362
 loglikelihood_stddev
 mo_mrm_objective_function_runoff, 363
 loglikelihood_trend_no_autocorr
 mo_mrm_objective_function_runoff, 365
 lookupintegral
 mo_neutrons, 564
 low_res_grid
 mo_common_variables::gridremapper, 777
 lower_bound
 mo_common_variables::gridremapper, 777
 lowres_id_on_highres

mo_common_variables::gridremapper, 777

mae_dp_1d

- mo_errormeasures, 148
- mo_errormeasures::mae, 793

mae_dp_2d

- mo_errormeasures, 148
- mo_errormeasures::mae, 793

mae_dp_3d

- mo_errormeasures, 149
- mo_errormeasures::mae, 794

mae_sp_1d

- mo_errormeasures, 149
- mo_errormeasures::mae, 794

mae_sp_2d

- mo_errormeasures, 150
- mo_errormeasures::mae, 794

mae_sp_3d

- mo_errormeasures, 150
- mo_errormeasures::mae, 794

majority_statistics

- mo_upscaling_operators, 701

mapcoordinates

- mo_grid, 188

mask

- mo_common_variables::grid, 774
- mo_mrm_write_fluxes_states::outputvariable, 866
- mo_write_fluxes_states::outputvariable, 861

max_surfresist

- mo_mpr_constants, 264

max_timers

- mo_timer, 699

maxgeounit

- mo_mpr_constants, 264

maximummonthlyflow

- mo_mrm_signatures, 409

maxlen

- mo_ncwrite, 491

maxnlcovers

- mo_common_constants, 91

maxnobasins

- mo_common_constants, 91

maxnogauges

- mo_mrm_constants, 315

maxnosoilhorizons

- mo_mpr_constants, 264

mcmc_dp

- mo_mcmc, 226
- mo_mcmc::mcmc, 798

mcmc_error_params

- mo_common_mhm_mrm_variables, 102

mcmc_opti

- mo_common_mhm_mrm_variables, 102

mcmc_stddev_dp

- mo_mcmc, 227
- mo_mcmc::mcmc_stddev, 802

mdds

- mo_dd, 140

mean_dp

- mo_moment, 259
- mo_moment::mean, 804
- mo_template, 686
- mo_template::mean, 803

mean_sp

- mo_moment, 259
- mo_moment::mean, 804
- mo_template, 686
- mo_template::mean, 803

median_dp

- mo_percentile, 608
- mo_percentile::median, 804

median_sp

- mo_percentile, 609
- mo_percentile::median, 804

mend

- mo_common_variables::period, 876

message

- mo_message, 229

message_text

- mo_message, 230

meteo_forcings_wrapper

- mo_meteo_forcings, 234

meteo_weights_wrapper

- mo_meteo_forcings, 236

mhmm

- mo_mhmm, 239

mhmm_details

- mo_common_variables, 119

mhmm_driver

- mhmm_driver.f90, 933

mhmm_driver.f90, 933

- mhmm_driver, 933

mhmm_eval

- mo_mhmm_eval, 249

mhmm_initialize

- mo_startup, 676

mhmm_papers.md, 936

mhmm_read_config

- mo_mhmm_read_config, 253

missing

- mo_nml, 571

mixed_central_moment_dp

- mo_moment, 259
- mo_moment::mixed_central_moment, 805

mixed_central_moment_sp

- mo_moment, 259
- mo_moment::mixed_central_moment, 805

mixed_central_moment_var_dp

- mo_moment, 260
- mo_moment::mixed_central_moment_var, 805

mixed_central_moment_var_sp

- mo_moment, 260
- mo_moment::mixed_central_moment_var, 806

mo_anneal, 75

- anneal_dp, 75
- dchange_dp, 76
- generate_neighborhood_weight_dp, 77

gettemperature_dp, 77
 pargen_anneal_dp, 78
 pargen_dds_dp, 78
 mo_anneal.f90, 936
 mo_anneal::anneal, 729
 anneal_dp, 731
 mo_anneal::generate_neighborhood_weight, 763
 generate_neighborhood_weight_dp, 763
 mo_anneal::gettemperature, 770
 gettemperature_dp, 771
 mo_append, 79
 append_char_3d, 80
 append_char_m_m, 80
 append_char_v_s, 81
 append_char_v_v, 81
 append_dp_3d, 81
 append_dp_m_m, 81
 append_dp_v_s, 81
 append_dp_v_v, 81
 append_i4_3d, 82
 append_i4_m_m, 82
 append_i4_v_s, 82
 append_i4_v_v, 82
 append_i8_3d, 82
 append_i8_m_m, 82
 append_i8_v_s, 82
 append_i8_v_v, 83
 append_lgt_3d, 83
 append_lgt_m_m, 83
 append_lgt_v_s, 83
 append_lgt_v_v, 83
 append_sp_3d, 83
 append_sp_m_m, 84
 append_sp_v_s, 84
 append_sp_v_v, 84
 paste_char_m_m, 84
 paste_char_m_s, 84
 paste_char_m_v, 84
 paste_dp_m_m, 84
 paste_dp_m_s, 85
 paste_dp_m_v, 85
 paste_i4_m_m, 85
 paste_i4_m_s, 85
 paste_i4_m_v, 85
 paste_i8_m_m, 85
 paste_i8_m_s, 85
 paste_i8_m_v, 86
 paste_lgt_m_m, 86
 paste_lgt_m_s, 86
 paste_lgt_m_v, 86
 paste_sp_m_m, 86
 paste_sp_m_s, 86
 paste_sp_m_v, 86
 mo_append.f90, 937
 mo_append::append, 732
 append_char_3d, 734
 append_char_m_m, 734
 append_char_v_s, 734
 append_char_v_v, 734
 append_dp_3d, 734
 append_dp_m_m, 735
 append_dp_v_s, 735
 append_dp_v_v, 735
 append_i4_m_m, 735
 append_i4_v_s, 735
 append_i4_v_v, 735
 append_i8_3d, 736
 append_i8_m_m, 736
 append_i8_v_s, 736
 append_i8_v_v, 736
 append_lgt_3d, 736
 append_lgt_m_m, 736
 append_lgt_v_s, 736
 append_lgt_v_v, 737
 append_sp_3d, 737
 append_sp_m_m, 737
 append_sp_v_s, 737
 append_sp_v_v, 737
 mo_append::paste, 868
 paste_char_m_m, 869
 paste_char_m_s, 869
 paste_char_m_v, 870
 paste_dp_m_m, 870
 paste_dp_m_s, 870
 paste_dp_m_v, 870
 paste_i4_m_m, 870
 paste_i4_m_s, 870
 paste_i4_m_v, 870
 paste_i8_m_m, 871
 paste_i8_m_s, 871
 paste_i8_m_v, 871
 paste_lgt_m_m, 871
 paste_lgt_m_s, 871
 paste_lgt_m_v, 871
 paste_sp_m_m, 872
 paste_sp_m_s, 872
 paste_sp_m_v, 872
 mo_canopy_interc, 87
 canopy_interc, 87
 mo_canopy_interc.f90, 938
 mo_common_constants, 89
 dayhours, 90
 daysecs, 90
 eps_dp, 90
 eps_sp, 90
 hoursecs, 90
 maxnlcovers, 91
 maxnobasins, 91
 ncolpars, 91
 nodata_dp, 91
 nodata_i4, 91
 p1_initstatefluxes, 92
 yeardays, 92
 yearmonths, 92
 yearmonths_i4, 92
 mo_common_constants.f90, 938

mo_common_file, 92
 file_config, 93
 file_dem, 93
 uconfig, 93
 udem, 93
 ulcoverclass, 94
mo_common_file.f90, 939
mo_common_functions, 94
 in_bound, 94
mo_common_functions.f90, 939
mo_common_mHM_mRM_file.f90, 940
mo_common_mHM_mRM_read_config.f90, 940
mo_common_mHM_mRM_variables.f90, 940
mo_common_mhm_mrm_file, 95
 file_opti, 95
 file_opti_nml, 96
 uopti, 96
 uopti_nml, 96
mo_common_mhm_mrm_read_config, 96
 check_optimization_settings, 97
 common_check_resolution, 98
 common_mhm_mrm_read_config, 98
mo_common_mhm_mrm_variables, 100
 dds_r, 101
 dirrestartin, 101
 evalper, 102
 lcyearid, 102
 mcmc_error_params, 102
 mcmc_opti, 102
 mrm_coupling_mode, 102
 nerror_model, 102
 niterations, 103
 ntstepday, 103
 opti_function, 103
 opti_method, 103
 optimize, 103
 optimize_restart, 103
 read_restart, 104
 readper, 104
 resolutionrouting, 104
 sa_temp, 104
 sce_ngs, 104
 sce_npg, 104
 sce_nps, 104
 seed, 105
 simper, 105
 timestep, 105
 warmingdays, 105
 warmper, 105
mo_common_read_config, 106
 common_read_config, 106
 set_land_cover_scenes_id, 107
mo_common_read_config.f90, 941
mo_common_read_data, 108
 read_dem, 109
 read_lccover, 110
mo_common_read_data.f90, 941
mo_common_restart, 111
 read_grid_info, 111
 write_grid_info, 113
mo_common_restart.f90, 942
mo_common_variables, 114
 alma_convention, 115
 contact, 116
 conventions, 116
 dircommonfiles, 116
 dirconfigout, 116
 dirlcover, 116
 dirmorpho, 116
 dirout, 116
 dirrestartout, 117
 filelatlon, 117
 global_parameters, 117
 global_parameters_name, 117
 history, 117
 iflag_coordinate_sys, 117
 lo_basin, 118
 lo_elev, 118
 lo_l1_remap, 118
 lo_lccover, 118
 lc_year_end, 118
 lc_year_start, 118
 lcfilename, 119
 level0, 119
 level1, 119
 mhm_details, 119
 nbasins, 119
 nlcoverscene, 120
 nprocesses, 120
 nuniquel0basins, 120
 processmatrix, 120
 project_details, 121
 resolutionhydrology, 121
 setup_description, 121
 simulation_type, 121
 write_restart, 121
mo_common_variables.f90, 942
mo_common_variables::grid, 773
 cellarea, 773
 cellcoor, 773
 cellsize, 774
 id, 774
 iend, 774
 istart, 774
 mask, 774
 ncells, 774
 ncols, 774
 nodata_value, 774
 nrows, 774
 x, 775
 xllcorner, 775
 y, 775
 yllcorner, 775
mo_common_variables::gridremapper, 776
 high_res_grid, 776
 left_bound, 777

low_res_grid, 777
 lower_bound, 777
 lowres_id_on_highres, 777
 n_subcells, 777
 right_bound, 777
 upper_bound, 777
 mo_common_variables::period, 875
 dend, 876
 dstart, 876
 julend, 876
 julstart, 876
 mend, 876
 mstart, 876
 nobs, 876
 yend, 876
 ystart, 877
 mo_constants, 121
 cp0_dp, 124
 cp0_sp, 124
 dayhours, 124
 daysecs, 124
 deg2rad_dp, 124
 deg2rad_sp, 124
 euler, 125
 euler_d, 125
 gravity_dp, 125
 gravity_sp, 125
 nerr, 125
 nin, 125
 nnml, 125
 nout, 126
 p0_dp, 126
 p0_sp, 126
 pi, 126
 pi_d, 126
 pi_dp, 126
 pi_sp, 126
 pio2, 127
 pio2_d, 127
 pio2_dp, 127
 pio2_sp, 127
 psychro_dp, 127
 psychro_sp, 127
 rad2deg_dp, 127
 rad2deg_sp, 127
 radiusearth_dp, 128
 radiusearth_sp, 128
 rho0_dp, 128
 rho0_sp, 128
 secday_dp, 128
 secday_sp, 128
 sigma_dp, 128
 sigma_sp, 128
 solarconst_dp, 129
 solarconst_sp, 129
 specheatet_dp, 129
 specheatet_sp, 129
 sqrt2, 129
 sqrt2_d, 129
 sqrt2_dp, 129
 sqrt2_sp, 130
 t0_dp, 130
 t0_sp, 130
 twopi, 130
 twopi_d, 130
 twopi_dp, 130
 twopi_sp, 130
 twothird_dp, 131
 twothird_sp, 131
 yeardays, 131
 yearmonths, 131
 mo_constants.f90, 943
 mo_corr, 131
 arth_dp, 132
 arth_i4, 132
 arth_sp, 133
 autocoeffk_1d_dp, 133
 autocoeffk_1d_sp, 133
 autocoeffk_dp, 133
 autocoeffk_sp, 133
 autocorr_1d_dp, 133
 autocorr_1d_sp, 134
 autocorr_dp, 134
 autocorr_sp, 134
 corr_dp, 134
 corr_sp, 134
 crosscoeffk_dp, 134
 crosscoeffk_sp, 135
 crosscorr_dp, 135
 crosscorr_sp, 135
 four1_dp, 135
 four1_sp, 135
 fourrow_dp, 135
 fourrow_sp, 136
 npar2_arth, 138
 npar_arth, 138
 realft_dp, 136
 realft_sp, 136
 swap_1d_dpc, 137
 swap_1d_spc, 137
 zroots_unity_dp, 137
 zroots_unity_sp, 137
 mo_corr.f90, 945
 mo_corr::arth, 737
 arth_dp, 738
 arth_i4, 738
 arth_sp, 738
 mo_corr::autocoeffk, 740
 autocoeffk_1d_dp, 740
 autocoeffk_1d_sp, 740
 autocoeffk_dp, 740
 autocoeffk_sp, 740
 mo_corr::autocorr, 741
 autocorr_1d_dp, 741
 autocorr_1d_sp, 741
 autocorr_dp, 741

autocorr_sp, 741
mo_corr::corr, 748
 corr_dp, 748
 corr_sp, 749
mo_corr::crosscoeffk, 750
 crosscoeffk_dp, 750
 crosscoeffk_sp, 750
mo_corr::crosscorr, 751
 crosscorr_dp, 751
 crosscorr_sp, 751
mo_corr::four1, 761
 four1_dp, 761
 four1_sp, 761
mo_corr::fourrow, 761
 fourrow_dp, 761
 fourrow_sp, 761
mo_corr::realft, 880
 realft_dp, 880
 realft_sp, 880
mo_corr::swap, 905
 swap_1d_dpc, 905
 swap_1d_spc, 905
mo_dds, 138
 dds, 139
 mdds, 140
 neigh_value, 142
mo_dds.f90, 946
mo_errormeasures, 142
 bias_dp_1d, 144
 bias_dp_2d, 144
 bias_dp_3d, 144
 bias_sp_1d, 144
 bias_sp_2d, 145
 bias_sp_3d, 145
 kge_dp_1d, 145
 kge_dp_2d, 145
 kge_dp_3d, 145
 kge_sp_1d, 145
 kge_sp_2d, 146
 kge_sp_3d, 146
 kgenocorr_dp_1d, 146
 kgenocorr_dp_2d, 146
 kgenocorr_dp_3d, 146
 kgenocorr_sp_1d, 146
 kgenocorr_sp_2d, 147
 kgenocorr_sp_3d, 147
 Innse_dp_1d, 147
 Innse_dp_2d, 147
 Innse_dp_3d, 147
 Innse_sp_1d, 147
 Innse_sp_2d, 147
 Innse_sp_3d, 148
 mae_dp_1d, 148
 mae_dp_2d, 148
 mae_dp_3d, 149
 mae_sp_1d, 149
 mae_sp_2d, 150
 mae_sp_3d, 150
 mse_dp_1d, 150
 mse_dp_2d, 151
 mse_dp_3d, 152
 mse_sp_1d, 152
 mse_sp_2d, 153
 mse_sp_3d, 154
 nse_dp_1d, 154
 nse_dp_2d, 155
 nse_dp_3d, 155
 nse_sp_1d, 155
 nse_sp_2d, 155
 nse_sp_3d, 155
 rmse_dp_1d, 155
 rmse_dp_2d, 156
 rmse_dp_3d, 156
 rmse_sp_1d, 156
 rmse_sp_2d, 157
 rmse_sp_3d, 157
 sae_dp_1d, 158
 sae_dp_2d, 158
 sae_dp_3d, 159
 sae_sp_1d, 160
 sae_sp_2d, 160
 sae_sp_3d, 161
 sse_dp_1d, 162
 sse_dp_2d, 162
 sse_dp_3d, 163
 sse_sp_1d, 164
 sse_sp_2d, 164
 sse_sp_3d, 165
 wnse_dp_1d, 166
 wnse_dp_2d, 166
 wnse_dp_3d, 166
 wnse_sp_1d, 166
 wnse_sp_2d, 166
 wnse_sp_3d, 166
mo_errormeasures.f90, 946
mo_errormeasures::bias, 744
 bias_dp_1d, 744
 bias_dp_2d, 745
 bias_dp_3d, 745
 bias_sp_1d, 745
 bias_sp_2d, 745
 bias_sp_3d, 745
mo_errormeasures::kge, 783
 kge_dp_1d, 784
 kge_dp_2d, 784
 kge_dp_3d, 785
 kge_sp_1d, 785
 kge_sp_2d, 785
 kge_sp_3d, 785
mo_errormeasures::kgenocorr, 785
 kgenocorr_dp_1d, 786
 kgenocorr_dp_2d, 786
 kgenocorr_dp_3d, 787
 kgenocorr_sp_1d, 787
 kgenocorr_sp_2d, 787
 kgenocorr_sp_3d, 787

mo_errormeasures::Innse, 790
 Innse_dp_1d, 791
 Innse_dp_2d, 791
 Innse_dp_3d, 791
 Innse_sp_1d, 791
 Innse_sp_2d, 791
 Innse_sp_3d, 791
 mo_errormeasures::mae, 793
 mae_dp_1d, 793
 mae_dp_2d, 793
 mae_dp_3d, 794
 mae_sp_1d, 794
 mae_sp_2d, 794
 mae_sp_3d, 794
 mo_errormeasures::mse, 808
 mse_dp_1d, 808
 mse_dp_2d, 809
 mse_dp_3d, 809
 mse_sp_1d, 809
 mse_sp_2d, 809
 mse_sp_3d, 809
 mo_errormeasures::nse, 846
 nse_dp_1d, 846
 nse_dp_2d, 846
 nse_dp_3d, 846
 nse_sp_1d, 846
 nse_sp_2d, 847
 nse_sp_3d, 847
 mo_errormeasures::rmse, 883
 rmse_dp_1d, 883
 rmse_dp_2d, 883
 rmse_dp_3d, 884
 rmse_sp_1d, 884
 rmse_sp_2d, 884
 rmse_sp_3d, 884
 mo_errormeasures::sae, 885
 sae_dp_1d, 885
 sae_dp_2d, 885
 sae_dp_3d, 886
 sae_sp_1d, 886
 sae_sp_2d, 886
 sae_sp_3d, 886
 mo_errormeasures::sse, 901
 sse_dp_1d, 902
 sse_dp_2d, 902
 sse_dp_3d, 902
 sse_sp_1d, 902
 sse_sp_2d, 902
 sse_sp_3d, 902
 mo_errormeasures::wnse, 929
 wnse_dp_1d, 929
 wnse_dp_2d, 929
 wnse_dp_3d, 929
 wnse_sp_1d, 929
 wnse_sp_2d, 929
 wnse_sp_3d, 930
 mo_file, 167
 file_defoutput, 167
 file_main, 168
 file_namelist_mhm, 168
 file_namelist_mhm_param, 168
 udefoutput, 168
 unamelist_mhm, 168
 unamelist_mhm_param, 168
 utws, 168
 version, 169
 version_date, 169
 mo_file.f90, 948
 mo_finish, 169
 finish, 169
 mo_finish.f90, 949
 mo_global_variables, 171
 basin_avg_tws_obs, 172
 basin_avg_tws_sim, 173
 dirabsvappressure, 173
 direvapotranspiration, 173
 dirmaxtemperature, 173
 dirmintemperature, 173
 dirnetradiation, 173
 dirneutrons, 173
 dirprecipitation, 174
 dirreferenceet, 174
 dirsoil_moisture, 174
 dirtemperature, 174
 dirwindspeed, 174
 evap_coeff, 174
 fday_pet, 174
 fday_prec, 175
 fday_temp, 175
 filetws, 175
 fnight_pet, 175
 fnight_prec, 175
 fnight_temp, 175
 inputformat_meteo_forcings, 175
 l1_absvappress, 176
 l1_aetc canopy, 176
 l1_aetsealed, 176
 l1_aetsoil, 176
 l1_baseflow, 176
 l1_et, 176
 l1_et_mask, 176
 l1_fastrunoff, 177
 l1_infilsoil, 177
 l1_inter, 177
 l1_melt, 177
 l1_netrad, 177
 l1_neutrons, 177
 l1_neutronsdata, 177
 l1_neutronsdata_mask, 178
 l1_percol, 178
 l1_pet, 178
 l1_pet_calc, 178
 l1_pet_weights, 178
 l1_pre, 178
 l1_pre_weights, 178
 l1_preeffect, 179

l1_rain, 179
l1_runoffseal, 179
l1_satstw, 179
l1_sealstw, 179
l1_slowrunoff, 179
l1_sm, 179
l1_sm_mask, 180
l1_snow, 180
l1_snowpack, 180
l1_soilmoist, 180
l1_temp, 180
l1_temp_weights, 180
l1_throughfall, 181
l1_tmax, 181
l1_tmin, 181
l1_total_runoff, 181
l1_unsatstw, 181
l1_windspeed, 181
level2, 181
neutron_integral_afast, 182
nmeasperday_tws, 182
nsoilhorizons_sm_input, 182
ntimesteps_l1_et, 182
ntimesteps_l1_neurons, 182
ntimesteps_l1_sm, 182
outputflxstate, 182
read_meteo_weights, 182
routingstates, 183
timestep_et_input, 183
timestep_model_inputs, 183
timestep_model_outputs, 183
timestep_neurons_input, 183
timestep_sm_input, 183
mo_global_variables.f90, 949
mo_global_variables::twsstructure, 907
 basinid, 907
 fname, 907
 tws, 908
mo_grid, 184
 calculate_grid_properties, 184
 geocoordinates, 186
 init_lowres_level, 186
 l0_grid_setup, 187
 mapcoordinates, 188
 set_basin_indices, 189
mo_grid.f90, 951
mo_init_states, 190
 variables_alloc, 190
 variables_default_init, 191
mo_init_states.f90, 951
mo_julian, 192
 caldat, 194
 caldat360, 196
 caldat365, 196
 caldatjulian, 197
 calendar, 220
 date2dec, 199
 date2dec360, 200
 date2dec365, 201
 date2decjulian, 203
 dec2date, 204
 dec2date360, 206
 dec2date365, 208
 dec2datejulian, 209
 julday, 211
 julday360, 212
 julday365, 213
 juldayjulian, 214
 ndays, 216
 ndyin, 217
 selectcalendar, 218
 setcalendarinteger, 219
 setcalendarstring, 220
mo_julian.f90, 952
mo_julian::setcalendar, 886
 setcalendarinteger, 887
 setcalendarstring, 887
mo_kind, 221
 dp, 222
 dpc, 222
 i1, 222
 i2, 222
 i4, 222
 i8, 223
 lgt, 223
 sp, 223
 spc, 223
mo_kind.f90, 953
mo_kind::sprs2_dp, 899
 irow, 899
 jcol, 899
 len, 900
 n, 900
 val, 900
mo_kind::sprs2_sp, 900
 irow, 901
 jcol, 901
 len, 901
 n, 901
 val, 901
mo_linfit, 224
 linfit_dp, 224
 linfit_sp, 224
mo_linfit.f90, 954
mo_linfit::linfit, 789
 linfit_dp, 790
 linfit_sp, 790
mo_mcmc, 225
 generatenewparameterset_dp, 225
 mcmc_dp, 226
 mcmc_stddev_dp, 227
 pargen_dp, 227
 pargennorm_dp, 228
mo_mcmc.f90, 954
mo_mcmc::mcmc, 794
 mcmc_dp, 798

mo_mcmc::mcmc_stddev, 798
 mcmc_stddev_dp, 802
 mo_message, 228
 message, 229
 message_text, 230
 mo_message.f90, 955
 mo_meteo_forcings, 230
 chunk_config, 231
 chunk_size, 232
 is_read, 233
 meteo_forcings_wrapper, 234
 meteo_weights_wrapper, 236
 prepare_meteo_forcings_data, 237
 mo_meteo_forcings.f90, 955
 mo_mhm, 238
 mhm, 239
 mo_mhm.f90, 955
 mo_mhm_constants, 244
 c1_initstatesm, 245
 cosmic_alpha, 245
 cosmic_bd, 246
 cosmic_l1, 246
 cosmic_l2, 246
 cosmic_l3, 246
 cosmic_l4, 246
 cosmic_n, 246
 cosmic_vwclat, 246
 desilets_a0, 247
 desilets_a1, 247
 desilets_a2, 247
 duffiedelta1, 247
 duffiedelta2, 247
 duffiedr, 247
 h2odens, 247
 harsamconst, 247
 noutflxstate, 248
 p2_initstatefluxes, 248
 p3_initstatefluxes, 248
 p4_initstatefluxes, 248
 p5_initstatefluxes, 248
 satpressureslope1, 248
 stboltzmann, 248
 tetens_c1, 248
 tetens_c2, 249
 tetens_c3, 249
 mo_mhm_constants.f90, 956
 mo_mhm_eval, 249
 mhm_eval, 249
 mo_mhm_eval.f90, 957
 mo_mhm_read_config, 253
 mhm_read_config, 253
 mo_mhm_read_config.f90, 957
 mo_moment, 256
 absdev_dp, 257
 absdev_sp, 257
 average_dp, 257
 average_sp, 257
 central_moment_dp, 257
 central_moment_sp, 257
 central_moment_var_dp, 258
 central_moment_var_sp, 258
 correlation_dp, 258
 correlation_sp, 258
 covariance_dp, 258
 covariance_sp, 258
 kurtosis_dp, 259
 kurtosis_sp, 259
 mean_dp, 259
 mean_sp, 259
 mixed_central_moment_dp, 259
 mixed_central_moment_sp, 259
 mixed_central_moment_var_dp, 260
 mixed_central_moment_var_sp, 260
 moment_dp, 260
 moment_sp, 260
 skewness_dp, 260
 skewness_sp, 261
 stddev_dp, 261
 stddev_sp, 261
 variance_dp, 261
 variance_sp, 261
 mo_moment.f90, 957
 mo_moment::absdev, 729
 absdev_dp, 729
 absdev_sp, 729
 mo_moment::average, 741
 average_dp, 742
 average_sp, 742
 mo_moment::central_moment, 746
 central_moment_dp, 746
 central_moment_sp, 746
 mo_moment::central_moment_var, 746
 central_moment_var_dp, 746
 central_moment_var_sp, 746
 mo_moment::correlation, 749
 correlation_dp, 749
 correlation_sp, 749
 mo_moment::covariance, 749
 covariance_dp, 750
 covariance_sp, 750
 mo_moment::kurtosis, 787
 kurtosis_dp, 788
 kurtosis_sp, 788
 mo_moment::mean, 804
 mean_dp, 804
 mean_sp, 804
 mo_moment::mixed_central_moment, 805
 mixed_central_moment_dp, 805
 mixed_central_moment_sp, 805
 mo_moment::mixed_central_moment_var, 805
 mixed_central_moment_var_dp, 805
 mixed_central_moment_var_sp, 806
 mo_moment::moment, 806
 moment_dp, 806
 moment_sp, 806
 mo_moment::skewness, 887

skewness_dp, 888
skewness_sp, 888
mo_moment::stddev, 904
 stddev_dp, 904
 stddev_sp, 904
mo_moment::variance, 928
 variance_dp, 928
 variance_sp, 928
mo_mpr_constants, 261
 bulkdens_orgmatter, 263
 c1_initstatesm, 263
 field_cap_c1, 263
 field_cap_c2, 263
 karman, 263
 ks_c, 263
 lai_factor_surfresi, 263
 lai_offset_surfresi, 264
 max_surfresist, 264
 maxgeourit, 264
 maxnosoilhorizons, 264
 nlcover_class, 264
 p2_initstatefluxes, 264
 p3_initstatefluxes, 264
 p4_initstatefluxes, 264
 p5_initstatefluxes, 265
 pwp_c, 265
 pwp_matpot_theta, 265
 vgenuchten_sandtresh, 265
 vgenuchtенн_c1, 265
 vgenuchtенн_c10, 265
 vgenuchtенн_c11, 265
 vgenuchtенн_c12, 266
 vgenuchtенн_c13, 266
 vgenuchtенн_c14, 266
 vgenuchtенн_c15, 266
 vgenuchtенн_c16, 266
 vgenuchtенн_c17, 266
 vgenuchtенн_c18, 266
 vgenuchtенн_c2, 266
 vgenuchtенн_c3, 267
 vgenuchtенн_c4, 267
 vgenuchtенн_c5, 267
 vgenuchtенн_c6, 267
 vgenuchtенн_c7, 267
 vgenuchtенн_c8, 267
 vgenuchtенн_c9, 267
 windmeasheight, 268
mo_mpr_constants.f90, 958
mo_mpr_eval, 268
 mpr_eval, 268
mo_mpr_eval.f90, 959
mo_mpr_file, 271
 file_aspect, 272
 file_geolut, 272
 file_hydrogeoclass, 272
 file_laiclass, 272
 file_lailut, 273
 file_main, 273
 file_meteo_binary_end, 273
 file_meteo_header, 273
 file_namelist_mpr, 273
 file_namelist_mpr_param, 273
 file_slope, 273
 file_soil_database, 274
 file_soil_database_1, 274
 file_soilclass, 274
 uaspect, 274
 ugeolut, 274
 uhydrogeoclass, 274
 ulaiclass, 275
 ulailut, 275
 umeteo, 275
 umeteo_header, 275
 unamelist_mpr, 275
 unamelist_mpr_param, 275
 uslope, 276
 usoil_database, 276
 usoilclass, 276
 version, 276
 version_date, 276
mo_mpr_file.f90, 960
mo_mpr_global_variables, 276
 c2tstu, 278
 dirgridded_lai, 278
 fracsealed_cityarea, 278
 geounitkar, 278
 geounitlist, 278
 horizondepth_mhm, 278
 iflag_soildb, 279
 inputformat_gridded_lai, 279
 l0_asp, 279
 l0_geounit, 279
 l0_gridded_lai, 279
 l0_slope, 279
 l0_slope_emp, 280
 l0_soilid, 280
 l1_aeroresist, 280
 l1_alpha, 280
 l1_deggday, 280
 l1_deggdayinc, 280
 l1_deggdaymax, 280
 l1_deggdaynopre, 281
 l1_fasp, 281
 l1_froots, 281
 l1_fsealed, 281
 l1_harsamcoeff, 281
 l1_jarvis_thresh_c1, 281
 l1_karstloss, 282
 l1_kbaseflow, 282
 l1_kfastflow, 282
 l1_kperco, 282
 l1_kslowflow, 282
 l1_maxinter, 282
 l1_petlaicorfactor, 282
 l1_prietaryalpha, 283
 l1_sealedthresh, 283

l1_soilmoistexp, 283
 l1_soilmoistfc, 283
 l1_soilmoistsat, 283
 l1_surfresist, 283
 l1_tempthresh, 283
 l1_unsatthresh, 284
 l1_wiltingpoint, 284
 laiut, 284
 laiper, 284
 laiunitlist, 284
 ngeounits, 284
 nlai, 284
 nlaiclass, 285
 nsoilhorizons_mhm, 285
 nsoiltypes, 285
 soildb, 285
 tillagedepth, 285
 timestep_lai_input, 285
 mo_mpr_global_variables.f90, 961
 mo_mpr_global_variables::soiltype, 888
 clay, 889
 db, 889
 dbm, 889
 depth, 889
 id, 889
 is_present, 889
 ks, 890
 ld, 890
 nhorizons, 890
 ntillhorizons, 890
 rzdepth, 890
 sand, 890
 thetafc, 890
 thetafc_till, 890
 thetapw, 890
 thetapw_till, 891
 thetas, 891
 thetas_till, 891
 ud, 891
 wd, 891
 mo_mpr_pet, 286
 bulksurface_resistance, 286
 pet_correctbyasp, 287
 pet_correctbylai, 288
 priestley_taylor_alpha, 290
 mo_mpr_pet.f90, 962
 mo_mpr_read_config, 291
 mpr_read_config, 292
 mo_mpr_read_config.f90, 963
 mo_mpr_restart, 293
 unpack_field_and_write_1d_dp, 294
 unpack_field_and_write_1d_i4, 294
 unpack_field_and_write_2d_dp, 294
 unpack_field_and_write_3d_dp, 295
 write_eff_params, 295
 write_mpr_restart_files, 296
 mo_mpr_restart.f90, 963
 mo_mpr_restart::unpack_field_and_write, 911
 unpack_field_and_write_1d_dp, 911
 unpack_field_and_write_1d_i4, 912
 unpack_field_and_write_2d_dp, 912
 unpack_field_and_write_3d_dp, 912
 mo_mpr_runoff, 297
 mpr_runoff, 298
 mo_mpr_runoff.f90, 963
 mo_mpr_smhorizons, 300
 mpr_smhorizons, 300
 mo_mpr_smhorizons.f90, 964
 mo_mpr_soilmoist, 302
 field_cap, 303
 genuchten, 304
 hydro_cond, 305
 mpr_sm, 306
 pwp, 308
 mo_mpr_soilmoist.f90, 964
 mo_mpr_startup, 309
 init_eff_params, 310
 l0_check_input, 310
 l0_variable_init, 311
 mpr_initialize, 313
 mo_mpr_startup.f90, 964
 mo_mrm_constants, 314
 deltah, 315
 given_ts, 315
 maxnogauges, 315
 noutflxstate, 315
 nroutingstates, 315
 rout_space_weight, 316
 mo_mrm_constants.f90, 965
 mo_mrm_eval, 316
 mrm_eval, 316
 mo_mrm_eval.f90, 965
 mo_mrm_file, 318
 file_config, 320
 file_daily_discharge, 320
 file_defoutput, 320
 file_facc, 320
 file_fdir, 320
 file_gaugeloc, 320
 file_main, 320
 file_mrm_output, 321
 file_namelist_mrm, 321
 file_namelist_param_mrm, 321
 ncfile_discharge, 321
 uconfig, 321
 udaily_discharge, 321
 udefoutput, 322
 udischarge, 322
 ufacc, 322
 ufdir, 322
 ugaugeloc, 322
 unamelist_mrm, 322
 unamelist_param_mrm, 322
 version, 323
 version_date, 323
 mo_mrm_file.f90, 965

mo_mrm_global_variables, 323
basin_mrm, 325
dirgauges, 325
dirtotalrunoff, 325
filenametotalrunoff, 325
gauge, 325
inflowgauge, 325
is_start, 326
I0_dracell, 326
I0_drasc, 326
I0_facc, 326
I0_fdir, 326
I0_floodplain, 326
I0_gaugeloc, 326
I0_inflowgaugeloc, 327
I0_l11_remap, 327
I0_streamnet, 327
I11_afloodplain, 327
I11_c1, 327
I11_c2, 327
I11_colout, 328
I11_fcol, 328
I11_fdir, 328
I11_fromn, 328
I11_frow, 328
I11_k, 328
I11_l1_id, 328
I11_label, 329
I11_length, 329
I11_netperm, 329
I11_nlinkfracfpimp, 329
I11_noutlets, 329
I11_qmod, 329
I11_qout, 330
I11_qtin, 330
I11_qtr, 330
I11_rorder, 330
I11_rowout, 330
I11_sink, 330
I11_slope, 331
I11_tcol, 331
I11_ton, 331
I11_trow, 331
I11_tsrouter, 331
I11_xi, 331
I1_l11_id, 332
I1_l11_remap, 332
I1_total_runoff_in, 332
level11, 332
mrm_runoff, 332
ngaugestotal, 332
ninfilegaugestotal, 333
nmeasperday, 333
outputflxstate_mrm, 333
timestep_model_outputs_mrm, 333
varnametotalrunoff, 333
mo_mrm_global_variables.f90, 966
mo_mrm_global_variables::basininfo_mrm, 742
gaugeidlist, 743
gaugeindexlist, 743
gaugenodelist, 743
inflowgaugeheadwater, 743
inflowgaugeidlist, 743
inflowgaugeindexlist, 743
inflowgaugenodelist, 743
I0_coloutlet, 744
I0_noutlet, 744
I0_rowoutlet, 744
ngauges, 744
ninfilegauges, 744
mo_mrm_global_variables::gaugingstation, 762
basinid, 762
fname, 762
gaugeid, 762
q, 763
mo_mrm_init, 333
config_output, 334
I0_check_input_routing, 335
mrm_init, 336
mrm_init_param, 339
mrm_update_param, 340
print_startup_message, 341
variables_alloc_routing, 341
variables_default_init_routing, 342
mo_mrm_init.f90, 968
mo_mrm_mpr, 343
reg_rout, 344
mo_mrm_mpr.f90, 968
mo_mrm_net_startup, 345
celllength, 346
get_distance_two_lat_lon_points, 347
I11_flow_direction, 348
I11_fraction_sealed_floodplain, 350
I11_l1_mapping, 351
I11_link_location, 352
I11_routing_order, 353
I11_set_drain_outlet_gauges, 354
I11_set_network_topology, 355
I11_stream_features, 356
movedownonecell, 357
moveup, 358
mo_mrm_net_startup.f90, 969
mo_mrm_objective_function_runoff, 359
extract_runoff, 360
loglikelihood_levin2013_2, 362
loglikelihood_stddev, 363
loglikelihood_trend_no_autocorr, 365
multi_objective_ae_fdc_lsv_nse_djf, 366
multi_objective_lnnse_highflow_lnnse_lowflow, 367
multi_objective_lnnse_highflow_lnnse_lowflow_2, 369
multi_objective_nse_lnnse, 370
multi_objective_runoff, 371
objective_equal_nse_lnnse, 372
objective_kge, 374

objective_lnnse, 375
 objective_multiple_gauges_kge_power6, 377
 objective_nse, 378
 objective_power6_nse_lnnse, 379
 objective_sse, 380
 objective_weighted_nse, 382
 parameter_regularization, 383
 single_objective_runoff, 384
 mo_mrm_objective_function_runoff.f90, 969
 mo_mrm_read_config, 385
 mrm_read_config, 386
 read_mrm_routing_params, 387
 mo_mrm_read_config.f90, 970
 mo_mrm_read_data, 389
 mrm_read_discharge, 389
 mrm_read_l0_data, 390
 mrm_read_total_runoff, 391
 rotate_fdir_variable, 392
 mo_mrm_read_data.f90, 971
 mo_mrm_restart, 393
 mrm_read_restart_config, 394
 mrm_read_restart_states, 395
 mrm_write_restart, 396
 mo_mrm_restart.f90, 971
 mo_mrm_routing, 397
 add_inflow, 398
 l11_routing, 399
 l11_runoff_acc, 401
 mrm_routing, 402
 mo_mrm_routing.f90, 971
 mo_mrm_signatures, 405
 autocorrelation, 406
 flowdurationcurve, 406
 limb_densities, 408
 maximummonthlyflow, 409
 moments, 410
 peakdistribution, 411
 runoffratio, 412
 zeroflowratio, 413
 mo_mrm_signatures.f90, 972
 mo_mrm_write, 414
 average_counter, 423
 day_counter, 424
 month_counter, 424
 mrm_write, 415
 mrm_write_optifile, 416
 mrm_write_optinamelist, 417
 mrm_write_output_fluxes, 419
 nc, 424
 write_configfile, 421
 write_daily_obs_sim_discharge, 422
 year_counter, 424
 mo_mrm_write.f90, 973
 mo_mrm_write_fluxes_states, 424
 close, 425
 createoutputfile, 426
 newoutputdataset, 427
 newoutputvariable, 428
 updatedataset, 429
 updatevariable, 430
 writetimestep, 431
 writevariableattributes, 431
 writevariabletimestep, 432
 mo_mrm_write_fluxes_states.f90, 973
 mo_mrm_write_fluxes_states::outputdataset, 851
 all, 852
 basin, 852
 close, 852
 count, 852
 counter, 853
 created, 853
 ibasin, 853
 id, 853
 nc, 853
 ncdataset, 853
 steps, 853
 store, 853
 time, 853
 to, 854
 updatedataset, 852
 variables, 854
 vars, 854
 write, 854
 writetimestep, 852
 written, 854
 mo_mrm_write_fluxes_states::outputvariable, 864
 average, 865
 avg, 865
 before, 865
 between, 866
 calls, 866
 contains, 866
 count, 866
 counter, 866
 data, 866
 mask, 866
 nc, 866
 ncdataset, 867
 number, 867
 of, 867
 reconstruct, 867
 store, 867
 the, 867
 to, 867
 updatevariable, 865, 868
 variable, 868
 which, 868
 writes, 868
 writevariabletimestep, 865
 writing, 868
 mo_multi_param_reg, 433
 aerodynamical_resistance, 433
 baseflow_param, 435
 canopy_intercept_param, 436
 iper_thres_runoff, 437
 karstic_layer, 438

mpr, 439
 snow_acc_melt_param, 444
mo_multi_param_reg.f90, 974
mo_ncread, 445
 check, 446
 get_info, 447
 get_ncdim, 449
 get_ncdimatt, 451
 get_ncvar_0d_dp, 452
 get_ncvar_0d_i1, 452
 get_ncvar_0d_i4, 453
 get_ncvar_0d_sp, 453
 get_ncvar_1d_dp, 454
 get_ncvar_1d_i1, 454
 get_ncvar_1d_i4, 455
 get_ncvar_1d_sp, 455
 get_ncvar_2d_dp, 456
 get_ncvar_2d_i1, 456
 get_ncvar_2d_i4, 457
 get_ncvar_2d_sp, 457
 get_ncvar_3d_dp, 458
 get_ncvar_3d_i1, 458
 get_ncvar_3d_i4, 459
 get_ncvar_3d_sp, 459
 get_ncvar_4d_dp, 460
 get_ncvar_4d_i1, 460
 get_ncvar_4d_i4, 461
 get_ncvar_4d_sp, 461
 get_ncvar_5d_dp, 462
 get_ncvar_5d_i1, 462
 get_ncvar_5d_i4, 463
 get_ncvar_5d_sp, 463
 get_ncvaratt, 464
 ncclose, 464
 ncopen, 465
mo_ncread.f90, 975
mo_ncread::get_ncvar, 764
 get_ncvar_0d_dp, 764
 get_ncvar_0d_i1, 764
 get_ncvar_0d_i4, 765
 get_ncvar_0d_sp, 765
 get_ncvar_1d_dp, 765
 get_ncvar_1d_i1, 765
 get_ncvar_1d_i4, 765
 get_ncvar_1d_sp, 766
 get_ncvar_2d_dp, 766
 get_ncvar_2d_i1, 766
 get_ncvar_2d_i4, 766
 get_ncvar_2d_sp, 766
 get_ncvar_3d_dp, 767
 get_ncvar_3d_i1, 767
 get_ncvar_3d_i4, 767
 get_ncvar_3d_sp, 767
 get_ncvar_4d_dp, 768
 get_ncvar_4d_i1, 768
 get_ncvar_4d_i4, 768
 get_ncvar_4d_sp, 768
 get_ncvar_5d_dp, 768
get_ncvar_5d_i1, 769
get_ncvar_5d_i4, 769
get_ncvar_5d_sp, 769
mo_ncwrite, 465
 check, 467
 close_ncdf, 468
 create_ncdf, 470
 dnc, 491
 dump_ncdf_1d_dp, 471
 dump_ncdf_1d_i4, 471
 dump_ncdf_1d_sp, 472
 dump_ncdf_2d_dp, 472
 dump_ncdf_2d_i4, 473
 dump_ncdf_2d_sp, 473
 dump_ncdf_3d_dp, 474
 dump_ncdf_3d_i4, 474
 dump_ncdf_3d_sp, 475
 dump_ncdf_4d_dp, 475
 dump_ncdf_4d_i4, 476
 dump_ncdf_4d_sp, 476
 dump_ncdf_5d_dp, 477
 dump_ncdf_5d_i4, 477
 dump_ncdf_5d_sp, 478
 gatt, 491
 maxlen, 491
 nattdim, 492
 ndims, 492
 ngatt, 492
 nmaxatt, 492
 nmaxdim, 492
 nvars, 492
 open_ncdf, 478
 v, 492
 var2nc_1d_dp, 480
 var2nc_1d_i4, 481
 var2nc_1d_sp, 482
 var2nc_2d_dp, 482
 var2nc_2d_i4, 483
 var2nc_2d_sp, 484
 var2nc_3d_dp, 484
 var2nc_3d_i4, 485
 var2nc_3d_sp, 486
 var2nc_4d_dp, 486
 var2nc_4d_i4, 487
 var2nc_4d_sp, 488
 var2nc_5d_dp, 488
 var2nc_5d_i4, 489
 var2nc_5d_sp, 490
 write_dynamic_ncdf, 490
 write_static_ncdf, 491
mo_ncwrite.f90, 976
mo_ncwrite::attribute, 739
 name, 739
 nvalues, 739
 values, 739
 xtype, 739
mo_ncwrite::dims, 753
 dimid, 754

len, 754
 name, 754
mo_ncwrite::dump_netcdf, 754
 dump_netcdf_1d_dp, 754
 dump_netcdf_1d_i4, 755
 dump_netcdf_1d_sp, 755
 dump_netcdf_2d_dp, 755
 dump_netcdf_2d_i4, 755
 dump_netcdf_2d_sp, 756
 dump_netcdf_3d_dp, 756
 dump_netcdf_3d_i4, 756
 dump_netcdf_3d_sp, 756
 dump_netcdf_4d_dp, 756
 dump_netcdf_4d_i4, 757
 dump_netcdf_4d_sp, 757
 dump_netcdf_5d_dp, 757
 dump_netcdf_5d_i4, 757
 dump_netcdf_5d_sp, 757
mo_ncwrite::var2nc, 916
 var2nc_1d_dp, 917
 var2nc_1d_i4, 918
 var2nc_1d_sp, 918
 var2nc_2d_dp, 918
 var2nc_2d_i4, 918
 var2nc_2d_sp, 919
 var2nc_3d_dp, 919
 var2nc_3d_i4, 919
 var2nc_3d_sp, 920
 var2nc_4d_dp, 920
 var2nc_4d_i4, 920
 var2nc_4d_sp, 921
 var2nc_5d_dp, 921
 var2nc_5d_i4, 921
 var2nc_5d_sp, 922
mo_ncwrite::variable, 923
 att, 924
 count, 924
 dimids, 924
 dimtypes, 924
 g0_b, 925
 g0_d, 925
 g0_f, 925
 g0_i, 925
 g1_b, 925
 g1_d, 925
 g1_f, 925
 g1_i, 925
 g2_b, 925
 g2_d, 926
 g2_f, 926
 g2_i, 926
 g3_b, 926
 g3_d, 926
 g3_f, 926
 g3_i, 926
 g4_b, 926
 g4_d, 926
 g4_f, 927
 g4_i, 927
 name, 927
 natt, 927
 ndims, 927
 nlvls, 927
 nsubs, 927
 start, 927
 unlimited, 927
 varid, 928
 wflag, 928
 xtype, 928
mo_netcdf, 493
 check, 496
 close, 496
 equalncdimensions, 497
 getdata1df32, 497
 getdata1df64, 497
 getdata1di16, 498
 getdata1di32, 498
 getdata1di64, 499
 getdata1di8, 499
 getdata2df32, 500
 getdata2df64, 500
 getdata2di16, 501
 getdata2di32, 501
 getdata2di64, 502
 getdata2di8, 502
 getdata3df32, 503
 getdata3df64, 503
 getdata3di16, 504
 getdata3di32, 504
 getdata3di64, 505
 getdata3di8, 505
 getdata4df32, 506
 getdata4df64, 506
 getdata4di16, 507
 getdata4di32, 507
 getdata4di64, 508
 getdata4di8, 508
 getdata5df32, 509
 getdata5df64, 509
 getdata5di16, 510
 getdata5di32, 510
 getdata5di64, 511
 getdata5di8, 511
 getdatascalarf32, 512
 getdatascalarf64, 512
 getdatascalari16, 513
 getdatascalari32, 513
 getdatascalari64, 514
 getdatascalari8, 514
 getdimensionbyid, 515
 getdimensionbyname, 515
 getdimensionlength, 515
 getdimensionname, 516
 getdtypefrominteger, 516
 getdtypefromstring, 517
 getglobalattributechar, 517

getglobalattributef32, 517
getglobalattributef64, 518
getglobalattributei16, 518
getglobalattributei32, 519
getglobalattributei64, 519
getglobalattributei8, 520
getnodimensions, 520
getnovariables, 520
getreaddatashape, 521
getunlimiteddimension, 522
getvariableattributechar, 523
getvariableattributef32, 523
getvariableattributef64, 524
getvariableattributei16, 524
getvariableattributei32, 524
getvariableattributei64, 525
getvariableattributei8, 525
getvariablebyname, 526
getvariableldimensions, 526
getvariableldtype, 526
getvariablefillvaluef32, 527
getvariablefillvaluef64, 527
getvariablefillvaluei16, 527
getvariablefillvaluei32, 527
getvariablefillvaluei64, 527
getvariablefillvaluei8, 527
getvariableids, 528
getvariablelename, 528
getvariables, 528
getvariablesshape, 528
hasattribute, 529
hasdimension, 529
hasvariable, 529
initncdataset, 529
initncdimension, 529
initncvariable, 530
isdatasetunlimited, 530
isunlimiteddimension, 530
isunlimitedvariable, 530
newncdataset, 530
newncdimension, 530
newncvariable, 531
setdata1df32, 531
setdata1df64, 531
setdata1di16, 532
setdata1di32, 532
setdata1di64, 533
setdata1di8, 533
setdata2df32, 534
setdata2df64, 534
setdata2di16, 535
setdata2di32, 535
setdata2di64, 536
setdata2di8, 536
setdata3df32, 537
setdata3df64, 537
setdata3di16, 538
setdata3di32, 538
setdata3di64, 539
setdata3di8, 539
setdata4df32, 540
setdata4df64, 540
setdata4di16, 541
setdata4di32, 541
setdata4di64, 542
setdata4di8, 542
setdata5df32, 543
setdata5df64, 543
setdata5di16, 544
setdata5di32, 544
setdata5di64, 545
setdata5di8, 545
setdatascalarf32, 546
setdatascalarf64, 546
setdatascalari16, 547
setdatascalari32, 547
setdatascalari64, 547
setdatascalari8, 548
setdimension, 548
setglobalattributechar, 549
setglobalattributef32, 549
setglobalattributef64, 549
setglobalattributei16, 550
setglobalattributei32, 550
setglobalattributei64, 551
setglobalattributei8, 551
setvariableattributechar, 551
setvariableattributef32, 552
setvariableattributef64, 552
setvariableattributei16, 553
setvariableattributei32, 553
setvariableattributei64, 553
setvariableattributei8, 554
setvariablefillvaluef32, 554
setvariablefillvaluef64, 554
setvariablefillvaluei16, 554
setvariablefillvaluei32, 555
setvariablefillvaluei64, 555
setvariablefillvaluei8, 555
setvariablewithids, 555
setvariablewithnames, 556
setvariablewithtypes, 557
mo_ncdf.f90, 977
mo_ncdf::ncdataset, 811
 close, 813
 dataset, 821
 file, 821
 filename, 821
 fname, 821
 getattribute, 813
 getdimension, 814
 getdimensionbyid, 814
 getdimensionbyname, 814
 getglobalattributechar, 815
 getglobalattributef32, 815
 getglobalattributef64, 815

getglobalattributei16, 815
getglobalattributei32, 815
getglobalattributei64, 815
getglobalattributei8, 815
getnovariables, 815
getunlimiteddimension, 815
getvariable, 816
getvariablebyname, 816
getvariableids, 816
getvariables, 817
hasdimension, 817
hasvariable, 817
id, 821
initncdataset, 818
isunlimited, 818
mode, 821
netcdf, 822
of, 822
open, 822
opened, 822
setattribute, 818
setdimension, 819
setglobalattributechar, 819
setglobalattributef32, 819
setglobalattributef64, 819
setglobalattributei16, 819
setglobalattributei32, 819
setglobalattributei64, 820
setglobalattributei8, 820
setvariable, 820
setvariablewithids, 821
setvariablewithnames, 821
setvariablewithtypes, 821
the, 822
mo_netcdf::ncdimension, 822
dimension, 841
getattribute, 826
getdata, 827
getdata1df32, 827
getdata1df64, 827
getdata1di16, 827
getdata1di32, 827
getdata1di64, 827
getdata1di8, 828
getdata2df32, 828
getdata2df64, 828
getdata2di16, 828
getdata2di32, 828
getdata2di64, 828
getdata2di8, 828
getdata3df32, 828
getdata3df64, 828
getdata3di16, 829
getdata3di32, 829
getdata3di64, 829
getdata3di8, 829
getdata4df32, 829
getdata4df64, 829
getdata4di16, 829
getdata4di32, 829
getdata4di64, 829
getdata4di8, 830
getdata5df32, 830
getdata5df64, 830
getdata5di16, 830
getdata5di32, 830
getdata5di64, 830
getdata5di8, 830
getdatascalarf32, 830
getdatascalarf64, 830
getdatascalari16, 831
getdatascalari32, 831
getdatascalari64, 831
getdatascalari8, 831
getdimensions, 831
getdtype, 831
getfillvalue, 831
getname, 832
getnodimensions, 832
getshape, 832
getvariableattributechar, 832
getvariableattributef32, 832
getvariableattributef64, 832
getvariableattributei16, 832
getvariableattributei32, 832
getvariableattributei64, 833
getvariableattributei8, 833
getvariablefillvaluef32, 833
getvariablefillvaluef64, 833
getvariablefillvaluei16, 833
getvariablefillvaluei32, 833
getvariablefillvaluei64, 833
getvariablefillvaluei8, 833
hasattribute, 833
id, 841
initncvariable, 834
isunlimited, 834
netcdf, 841
parent, 841
s, 841
setattribute, 834
setdata, 834
setdata1df32, 835
setdata1df64, 835
setdata1di16, 835
setdata1di32, 835
setdata1di64, 835
setdata1di8, 835
setdata2df32, 836
setdata2df64, 836
setdata2di16, 836
setdata2di32, 836
setdata2di64, 836
setdata2di8, 836
setdata3df32, 836
setdata3df64, 836

setdata3di16, 836
setdata3di32, 837
setdata3di64, 837
setdata3di8, 837
setdata4df32, 837
setdata4df64, 837
setdata4di16, 837
setdata4di32, 837
setdata4di64, 837
setdata4di8, 837
setdata5df32, 838
setdata5df64, 838
setdata5di16, 838
setdata5di32, 838
setdata5di64, 838
setdata5di8, 838
setdatascalarf32, 838
setdatascalarf64, 838
setdatascalari16, 838
setdatascalari32, 839
setdatascalari64, 839
setdatascalari8, 839
setfillvalue, 839
setvariableattributechar, 839
setvariableattributef32, 839
setvariableattributef64, 840
setvariableattributei16, 840
setvariableattributei32, 840
setvariableattributei64, 840
setvariableattributei8, 840
setvariablefillvaluef32, 840
setvariablefillvaluef64, 840
setvariablefillvaluei16, 840
setvariablefillvaluei32, 840
setvariablefillvaluei64, 841
setvariablefillvaluei8, 841
the, 841, 842
mo_ncdf::ncvariable, 842
mo_neutrons, 557
approx_mon_int, 558
approx_mon_int_eps, 559
approx_mon_int_steps, 560
cosmic, 561
desiletsn0, 562
intgrandfast, 563
lookupintegral, 564
oldintegration, 564
tabularintegralfast, 565
mo_neutrons.f90, 980
mo_nml, 566
close_nml, 567
length_error, 571
missing, 571
nunitnml, 571
open_nml, 568
position_nml, 569
positioned, 571
read_error, 571
mo_nml.f90, 981
mo_objective_function, 572
extract_basin_avg_tws, 573
objective, 574
objective_et_kge_catchment_avg, 576
objective_kge_q_et, 577
objective_kge_q_rmse_et, 578
objective_kge_q_rmse_tws, 579
objective_kge_q_sm_corr, 581
objective_neutrons_kge_catchment_avg, 582
objective_sm_corr, 583
objective_sm_kge_catchment_avg, 584
objective_sm_pd, 586
objective_sm_sse_standard_score, 587
mo_objective_function.f90, 981
mo_optimization, 588
optimization, 589
mo_optimization.f90, 982
mo_optimization_utils, 590
mo_optimization_utils.f90, 982
mo_optimization_utils::eval_interface, 759
eval_interface, 760
mo_optimization_utils::objective_interface, 850
objective_interface, 850
mo_orderpack, 591
d_ctrper, 593
d_fndnth, 593
d_indmed, 593
d_indnth, 594
d_inspar, 594
d_inssor, 594
d_med, 594
d_median, 595
d_mrgref, 595
d_mrgrnk, 595
d_mulcnt, 595
d_nearless, 595
d_rapknr, 595
d_refpar, 596
d_refsor, 596
d_rinpar, 596
d_rnkpar, 596
d_subSOR, 596
d_uniinv, 597
d_unipar, 597
d_unirnk, 597
d_unista, 597
d_valmed, 597
d_valnth, 598
i_ctrper, 598
i_fndnth, 598
i_indmed, 598
i_indnth, 598
i_inspar, 599
i_inssor, 599
i_med, 599
i_median, 599
i_mrgref, 600

i_mrgrnk, 600
 i_mulcnt, 600
 i_nearless, 600
 i_rapknr, 600
 i_refpar, 600
 i_refsor, 600
 i_rinpar, 601
 i_rnkpar, 601
 i_subsov, 601
 i_uniinv, 602
 i_unipar, 602
 i_unirnk, 602
 i_unista, 602
 i_valmed, 602
 i_valnth, 602
 idont, 608
 r_ctrper, 603
 r_fndnth, 603
 r_indmed, 603
 r_indnth, 603
 r_inspars, 603
 r_inssor, 604
 r_med, 604
 r_median, 604
 r_mrgref, 604
 r_mrgrnk, 605
 r_mulcnt, 605
 r_nearless, 605
 r_rapknr, 605
 r_refpar, 605
 r_refsor, 605
 r_rinpar, 606
 r_rnkpar, 606
 r_subsov, 606
 r_uniinv, 606
 r_unipar, 607
 r_unirnk, 607
 r_unista, 607
 r_valmed, 607
 r_valnth, 607
 sort_index_dp, 607
 sort_index_i4, 607
 sort_index_sp, 608
 mo_orderpack.f90, 983
 mo_orderpack::ctrper, 751
 d_ctrper, 752
 i_ctrper, 752
 r_ctrper, 752
 mo_orderpack::fndnth, 760
 d_fndnth, 760
 i_fndnth, 760
 r_fndnth, 760
 mo_orderpack::indmed, 778
 d_indmed, 779
 i_indmed, 779
 r_indmed, 779
 mo_orderpack::indnth, 779
 d_indnth, 779
 i_indnth, 779
 r_indnth, 780
 mo_orderpack::inspar, 780
 d_inspars, 780
 i_inspars, 780
 r_inspars, 780
 mo_orderpack::inssor, 781
 d_inssor, 781
 i_inssor, 781
 r_inssor, 781
 mo_orderpack::mrgref, 807
 d_mrgref, 807
 i_mrgref, 807
 r_mrgref, 807
 mo_orderpack::mrgrnk, 807
 d_mrgrnk, 808
 i_mrgrnk, 808
 r_mrgrnk, 808
 mo_orderpack::mulcnt, 810
 d_mulcnt, 810
 i_mulcnt, 810
 r_mulcnt, 810
 mo_orderpack::nearless, 843
 d_nearless, 843
 i_nearless, 843
 r_nearless, 843
 mo_orderpack::omedian, 850
 d_median, 850
 i_median, 850
 r_median, 850
 mo_orderpack::rapknr, 877
 d_rapknr, 877
 i_rapknr, 878
 r_rapknr, 878
 mo_orderpack::refpar, 881
 d_refpar, 881
 i_refpar, 881
 r_refpar, 881
 mo_orderpack::refsor, 882
 d_refsor, 882
 i_refsor, 882
 r_refsor, 882
 mo_orderpack::rinpar, 882
 d_rinpar, 882
 i_rinpar, 883
 r_rinpar, 883
 mo_orderpack::rnkpar, 884
 d_rnkpar, 884
 i_rnkpar, 885
 r_rnkpar, 885
 mo_orderpack::sort, 891
 d_refsor, 894
 i_refsor, 894
 r_refsor, 894
 mo_orderpack::sort_index, 894
 sort_index_dp, 894
 sort_index_i4, 895
 sort_index_sp, 895

mo_orderpack::uniinv, 908
d_uniinv, 908
i_uniinv, 908
r_uniinv, 908
mo_orderpack::unipar, 909
d_unipar, 909
i_unipar, 909
r_unipar, 909
mo_orderpack::unirnk, 909
d_unirnk, 909
i_unirnk, 910
r_unirnk, 910
mo_orderpack::unista, 910
d_unista, 910
i_unista, 910
r_unista, 910
mo_orderpack::valmed, 914
d_valmed, 914
i_valmed, 914
r_valmed, 915
mo_orderpack::valnth, 915
d_valnth, 915
i_valnth, 915
r_valnth, 915
mo_percentile, 608
median_dp, 608
median_sp, 609
n_element_dp, 609
n_element_sp, 609
percentile_0d_dp, 609
percentile_0d_sp, 609
percentile_1d_dp, 610
percentile_1d_sp, 610
qmedian_dp, 610
qmedian_sp, 610
mo_percentile.f90, 985
mo_percentile::median, 804
median_dp, 804
median_sp, 804
mo_percentile::n_element, 810
n_element_dp, 810
n_element_sp, 811
mo_percentile::percentile, 874
percentile_0d_dp, 874
percentile_0d_sp, 874
percentile_1d_dp, 874
percentile_1d_sp, 875
mo_percentile::qmedian, 877
qmedian_dp, 877
qmedian_sp, 877
mo_pet, 610
extraterr_rad_approx, 611
pet_hargreaves, 612
pet_penman, 613
pet_priestly, 615
sat_vap_pressure, 616
slope_satpressure, 617
mo_pet.f90, 985
mo_prepare_gridded_lai, 618
prepare_gridded_daily_lai_data, 619
prepare_gridded_mean_monthly_lai_data, 620
mo_prepare_gridded_lai.f90, 986
mo_read_forcing_nc, 621
get_time_vector_and_select, 622
read_forcing_nc, 623
read_weights_nc, 625
mo_read_forcing_nc.f90, 986
mo_read_latlon, 626
read_latlon, 627
mo_read_latlon.f90, 987
mo_read_lut, 628
read_geoformation_lut, 629
read_lai_lut, 630
mo_read_lut.f90, 987
mo_read_optional_data, 631
read_basin_avg_tws, 631
read_evapotranspiration, 632
read_neutrons, 633
read_soil_moisture, 634
mo_read_optional_data.f90, 987
mo_read_spatial_data, 635
read_header_ascii, 636
read_spatial_data_ascii_dp, 637
read_spatial_data_ascii_i4, 638
mo_read_spatial_data.f90, 988
mo_read_spatial_data::read_spatial_data_ascii, 878
read_spatial_data_ascii_dp, 879
read_spatial_data_ascii_i4, 879
mo_read_timeseries, 639
read_timeseries, 639
mo_read_timeseries.f90, 988
mo_read_wrapper, 641
check_consistency_lut_map, 642
read_data, 643
mo_read_wrapper.f90, 988
mo_restart, 644
read_restart_states, 645
unpack_field_and_write_1d_dp, 646
unpack_field_and_write_1d_i4, 646
unpack_field_and_write_2d_dp, 647
unpack_field_and_write_3d_dp, 647
write_restart_files, 647
mo_restart.f90, 989
mo_restart::unpack_field_and_write, 912
unpack_field_and_write_1d_dp, 913
unpack_field_and_write_1d_i4, 913
unpack_field_and_write_2d_dp, 914
unpack_field_and_write_3d_dp, 914
mo_runoff, 648
l1_total_runoff, 649
runoff_sat_zone, 650
runoff_unsat_zone, 650
mo_runoff.f90, 989
mo_sce, 652
cce, 652
chkcst, 653

comp, 653
 getpnt, 654
 parstt, 655
 sce, 655
 sort_matrix, 659
 mo_sce.f90, 989
 set_optional, 990
 write_best_final, 990
 write_best_intermediate, 991
 write_population, 991
 write_termination_case, 991
 mo_set_netcdf_outputs, 660
 set_netcdf, 660
 mo_set_netcdf_outputs.f90, 992
 mo_snow_accum_melt, 661
 snow_accum_melt, 661
 mo_snow_accum_melt.f90, 992
 mo_soil_database, 662
 generate_soil_database, 663
 read_soil_lut, 664
 mo_soil_database.f90, 992
 mo_soil_moisture, 665
 feddes_et_reduction, 665
 jarvis_et_reduction, 666
 soil_moisture, 667
 mo_soil_moisture.f90, 993
 mo_spatial_agg_disagg_forcing, 669
 spatial_aggregation_3d, 670
 spatial_aggregation_4d, 670
 spatial_disaggregation_3d, 670
 spatial_disaggregation_4d, 671
 mo_spatial_agg_disagg_forcing.f90, 993
 mo_spatial_agg_disagg_forcing::spatial_aggregation, 895
 spatial_aggregation_3d, 895
 spatial_aggregation_4d, 896
 mo_spatial_agg_disagg_forcing::spatial_disaggregation, 896
 spatial_disaggregation_3d, 897
 spatial_disaggregation_4d, 897
 mo_spatialsimilarity, 671
 nndv_dp, 672
 nndv_sp, 672
 pd_dp, 672
 pd_sp, 672
 mo_spatialsimilarity.f90, 994
 mo_spatialsimilarity::nndv, 843
 nndv_dp, 845
 nndv_sp, 845
 mo_spatialsimilarity::pd, 872
 pd_dp, 873
 pd_sp, 874
 mo_standard_score, 672
 classified_standard_score_dp, 673
 classified_standard_score_sp, 673
 standard_score_dp, 673
 standard_score_sp, 674
 mo_standard_score.f90, 994
 mo_standard_score::classified_standard_score, 747
 classified_standard_score_dp, 748
 classified_standard_score_sp, 748
 mo_standard_score::standard_score, 903
 standard_score_dp, 904
 standard_score_sp, 904
 mo_startup, 674
 constants_init, 674
 l2_variable_init, 675
 mhmm_initialize, 676
 mo_startup.f90, 994
 mo_string_utils, 678
 compress, 678
 divide_string, 679
 dp2str, 680
 equalstrings, 680
 i42str, 680
 i4array2str, 681
 i82str, 681
 log2str, 681
 nonull, 681
 separator, 684
 sp2str, 682
 splitstring, 682
 startswith, 682
 str2num, 683
 tolower, 683
 toupper, 684
 mo_string_utils.f90, 995
 mo_string_utils::num2str, 847
 dp2str, 848
 i42str, 848
 i82str, 848
 log2str, 848
 sp2str, 848
 mo_string_utils::numarray2str, 849
 i4array2str, 849
 mo_template, 685
 circum, 685
 itest, 686
 mean_dp, 686
 mean_sp, 686
 pi_dp, 686
 pi_sp, 687
 mo_template.f90, 996
 mo_template::mean, 802
 mean_dp, 803
 mean_sp, 803
 mo_temporal_aggregation, 687
 day2mon_average_dp, 687
 hour2day_average_dp, 688
 mo_temporal_aggregation.f90, 996
 mo_temporal_aggregation::day2mon_average, 752
 day2mon_average_dp, 753
 mo_temporal_aggregation::hour2day_average, 777
 hour2day_average_dp, 778
 mo_temporal_disagg_forcing, 688
 temporal_disagg_forcing, 689

mo_temporal_disagg_forcing.f90, 997
mo_timer, 690
 clock_rate, 698
 cputime, 698
 cycles1, 698
 cycles2, 698
 cycles_max, 699
 max_timers, 699
 status, 699
 timer_check, 691
 timer_clear, 692
 timer_get, 693
 timer_print, 694
 timer_start, 695
 timer_stop, 696
 timers_init, 697
mo_timer.f90, 997
mo_upscaling_operators, 699
 l0_fractionalcover_in_lx, 700
 majority_statistics, 701
 upscale_arithmetic_mean, 701
 upscale_geometric_mean, 703
 upscale_harmonic_mean, 704
 upscale_p_norm, 705
mo_upscaling_operators.f90, 998
mo_utils, 706
 equal_dp, 707
 equal_sp, 707
 greaterequal_dp, 707
 greaterequal_sp, 707
 is_finite_dp, 708
 is_finite_sp, 708
 is_nan_dp, 708
 is_nan_sp, 708
 is_normal_dp, 708
 is_normal_sp, 708
 lesserequal_dp, 708
 lesserequal_sp, 708
 locate_0d_dp, 709
 locate_0d_sp, 709
 locate_1d_dp, 709
 locate_1d_sp, 709
 notequal_dp, 709
 notequal_sp, 709
 special_value_dp, 709
 special_value_sp, 710
 swap_vec_dp, 710
 swap_vec_i4, 710
 swap_vec_sp, 711
 swap_xy_dp, 711
 swap_xy_i4, 711
 swap_xy_sp, 711
mo_utils.f90, 998
mo_utils::eq, 758
 equal_dp, 758
 equal_sp, 758
mo_utils::equal, 758
 equal_dp, 759
 equal_sp, 759
 greaterequal_dp, 763
 greaterequal_sp, 763
 mo_utils::greaterequal, 772
 greaterequal_dp, 772
 greaterequal_sp, 772
 mo_utils::is_finite, 781
 is_finite_dp, 782
 is_finite_sp, 782
 mo_utils::is_nan, 782
 is_nan_dp, 782
 is_nan_sp, 782
 mo_utils::is_normal, 783
 is_normal_dp, 783
 is_normal_sp, 783
 mo_utils::le, 788
 lesserequal_dp, 788
 lesserequal_sp, 788
 mo_utils::lesserequal, 789
 lesserequal_dp, 789
 lesserequal_sp, 789
 mo_utils::locate, 792
 locate_0d_dp, 792
 locate_0d_sp, 792
 locate_1d_dp, 793
 locate_1d_sp, 793
 mo_utils::ne, 842
 notequal_dp, 842
 notequal_sp, 842
 mo_utils::notequal, 845
 notequal_dp, 845
 notequal_sp, 845
 mo_utils::special_value, 897
 special_value_dp, 898
 special_value_sp, 898
 mo_utils::swap, 905
 swap_vec_dp, 906
 swap_vec_i4, 906
 swap_vec_sp, 906
 swap_xy_dp, 906
 swap_xy_i4, 906
 swap_xy_sp, 907
 mo_write_ascii, 711
 write_configfile, 712
 write_optifile, 713
 write_optinamelist, 714
 mo_write_ascii.f90, 999
 mo_write_fluxes_states, 715
 close, 716
 createoutputfile, 716
 fluxesunit, 717
 newoutputdataset, 718
 newoutputvariable, 719
 updatedataset, 720
 updatevariable, 721
 writetimestep, 722
 writevariableattributes, 723

writevariabletimestep, 723
 mo_write_fluxes_states.f90, 1000
 mo_write_fluxes_states::outputdataset, 855
 all, 856
 basin, 856
 close, 856
 count, 856
 counter, 857
 created, 857
 ibasin, 857
 id, 857
 nc, 857
 ncdataset, 857
 steps, 857
 store, 857
 time, 857
 to, 858
 updatedataset, 856
 variables, 858
 vars, 858
 write, 858
 writetimestep, 856
 written, 858
 mo_write_fluxes_states::outputvariable, 859
 average, 860
 avg, 860
 before, 860
 between, 861
 calls, 861
 contains, 861
 count, 861
 counter, 861
 data, 861
 mask, 861
 nc, 861
 ncdataset, 862
 number, 862
 of, 862
 reconstruct, 862
 store, 862
 the, 862
 to, 862
 updatevariable, 860, 863
 variable, 863
 which, 863
 writes, 863
 writevariabletimestep, 860
 writing, 863
 mo_xor4096, 724
 get_timeseed_i4_0d, 725
 get_timeseed_i4_1d, 725
 get_timeseed_i8_0d, 725
 get_timeseed_i8_1d, 725
 n_save_state, 728
 xor4096d_0d, 725
 xor4096d_1d, 725
 xor4096f_0d, 726
 xor4096f_1d, 726
 xor4096gd_0d, 726
 xor4096gd_1d, 726
 xor4096gf_0d, 726
 xor4096gf_1d, 727
 xor4096l_0d, 727
 xor4096l_1d, 727
 xor4096s_0d, 727
 xor4096s_1d, 727
 mo_xor4096.f90, 1001
 mo_xor4096::get_timeseed, 769
 get_timeseed_i4_0d, 769
 get_timeseed_i4_1d, 770
 get_timeseed_i8_0d, 770
 get_timeseed_i8_1d, 770
 mo_xor4096::xor4096, 930
 xor4096d_0d, 930
 xor4096d_1d, 930
 xor4096f_0d, 930
 xor4096f_1d, 931
 xor4096l_0d, 931
 xor4096l_1d, 931
 xor4096s_0d, 931
 xor4096s_1d, 931
 mo_xor4096::xor4096g, 932
 xor4096gd_0d, 932
 xor4096gd_1d, 932
 xor4096gf_0d, 932
 xor4096gf_1d, 932
 mode
 mo_ncdf::ncdataset, 821
 moment_dp
 mo_moment, 260
 mo_moment::moment, 806
 moment_sp
 mo_moment, 260
 mo_moment::moment, 806
 moments
 mo_mrm_signatures, 410
 month_counter
 mo_mrm_write, 424
 movedownonecell
 mo_mrm_net_startup, 357
 moveup
 mo_mrm_net_startup, 358
 mpr
 mo_multi_param_reg, 439
 mpr_driver
 mpr_driver.f90, 1001
 mpr_driver.f90, 1001
 mpr_driver, 1001
 mpr_eval
 mo_mpr_eval, 268
 mpr_initialize
 mo_mpr_startup, 313
 mpr_read_config
 mo_mpr_read_config, 292
 mpr_runoff
 mo_mpr_runoff, 298

mpr_sm
 mo_mpr_soilmoist, 306
mpr_smhorizons
 mo_mpr_smhorizons, 300
mrm_coupling_mode
 mo_common_mhm_mrm_variables, 102
mrm_driver
 mrm_driver.f90, 1004
mrm_driver.f90, 1003
 mrm_driver, 1004
mrm_eval
 mo_mrm_eval, 316
mrm_init
 mo_mrm_init, 336
mrm_init_param
 mo_mrm_init, 339
mrm_read_config
 mo_mrm_read_config, 386
mrm_read_discharge
 mo_mrm_read_data, 389
mrm_read_l0_data
 mo_mrm_read_data, 390
mrm_read_restart_config
 mo_mrm_restart, 394
mrm_read_restart_states
 mo_mrm_restart, 395
mrm_read_total_runoff
 mo_mrm_read_data, 391
mrm_routing
 mo_mrm_routing, 402
mrm_runoff
 mo_mrm_global_variables, 332
mrm_update_param
 mo_mrm_init, 340
mrm_write
 mo_mrm_write, 415
mrm_write_optifile
 mo_mrm_write, 416
mrm_write_optinamelist
 mo_mrm_write, 417
mrm_write_output_fluxes
 mo_mrm_write, 419
mrm_write_restart
 mo_mrm_restart, 396
mse_dp_1d
 mo_errormeasures, 150
 mo_errormeasures::mse, 808
mse_dp_2d
 mo_errormeasures, 151
 mo_errormeasures::mse, 809
mse_dp_3d
 mo_errormeasures, 152
 mo_errormeasures::mse, 809
mse_sp_1d
 mo_errormeasures, 152
 mo_errormeasures::mse, 809
mse_sp_2d
 mo_errormeasures, 153
 mo_errormeasures::mse, 809
mse_sp_3d
 mo_errormeasures, 154
 mo_errormeasures::mse, 809
mstart
 mo_common_variables::period, 876
multi_objective_ae_fdc_lsv_nse_djf
 mo_mrm_objective_function_runoff, 366
multi_objective_lnnse_highflow_lnnse_lowflow
 mo_mrm_objective_function_runoff, 367
multi_objective_lnnse_highflow_lnnse_lowflow_2
 mo_mrm_objective_function_runoff, 369
multi_objective_nse_lnnse
 mo_mrm_objective_function_runoff, 370
multi_objective_runoff
 mo_mrm_objective_function_runoff, 371
n
 mo_kind::sprs2_dp, 900
 mo_kind::sprs2_sp, 901
n_element_dp
 mo_percentile, 609
 mo_percentile::n_element, 810
n_element_sp
 mo_percentile, 609
 mo_percentile::n_element, 811
n_save_state
 mo_xor4096, 728
n_subcells
 mo_common_variables::gridremapper, 777
name
 mo_ncwrite::attribute, 739
 mo_ncwrite::dims, 754
 mo_ncwrite::variable, 927
natt
 mo_ncwrite::variable, 927
nattdim
 mo_ncwrite, 492
nbasins
 mo_common_variables, 119
nc
 mo_mrm_write, 424
 mo_mrm_write_fluxes_states::outputdataset, 853
 mo_mrm_write_fluxes_states::outputvariable, 866
 mo_write_fluxes_states::outputdataset, 857
 mo_write_fluxes_states::outputvariable, 861
ncclose
 mo_ncread, 464
ncdataset
 mo_mrm_write_fluxes_states::outputdataset, 853
 mo_mrm_write_fluxes_states::outputvariable, 867
 mo_write_fluxes_states::outputdataset, 857
 mo_write_fluxes_states::outputvariable, 862
ncells
 mo_common_variables::grid, 774
ncfile_discharge
 mo_mrm_file, 321
ncolpars
 mo_common_constants, 91

ncols
 mo_common_variables::grid, 774
 ncopen
 mo_ncread, 465
 ndays
 mo Julian, 216
 ndims
 mo_ncwrite, 492
 mo_ncwrite::variable, 927
 ndyin
 mo Julian, 217
 neigh_value
 mo_dds, 142
 nerr
 mo_constants, 125
 nerror_model
 mo_common_mhm_mrm_variables, 102
 netcdf
 mo_netcdf::ncdataset, 822
 mo_netcdf::ncdimension, 841
 neutron_integral_afast
 mo_global_variables, 182
 newncdataset
 mo_netcdf, 530
 newncdimension
 mo_netcdf, 530
 newncvariable
 mo_netcdf, 531
 newoutputdataset
 mo_mrm_write_fluxes_states, 427
 mo_write_fluxes_states, 718
 newoutputvariable
 mo_mrm_write_fluxes_states, 428
 mo_write_fluxes_states, 719
 ngatt
 mo_ncwrite, 492
 ngauges
 mo_mrm_global_variables::basininfo_mrm, 744
 ngaugestotal
 mo_mrm_global_variables, 332
 ngeounits
 mo_mpr_global_variables, 284
 nhorizons
 mo_mpr_global_variables::soiltype, 890
 nin
 mo_constants, 125
 ninflowgauges
 mo_mrm_global_variables::basininfo_mrm, 744
 ninflowgaugestotal
 mo_mrm_global_variables, 333
 niterations
 mo_common_mhm_mrm_variables, 103
 nlai
 mo_mpr_global_variables, 284
 nlaiclass
 mo_mpr_global_variables, 285
 nlcover_class
 mo_mpr_constants, 264
 nlcoverscene
 mo_common_variables, 120
 nlvls
 mo_ncwrite::variable, 927
 nmaxatt
 mo_ncwrite, 492
 nmaxdim
 mo_ncwrite, 492
 nmeasperday
 mo_mrm_global_variables, 333
 nmeasperday_tws
 mo_global_variables, 182
 nndv_dp
 mo_spatialssimilarity, 672
 mo_spatialssimilarity::nndv, 845
 nndv_sp
 mo_spatialssimilarity, 672
 mo_spatialssimilarity::nndv, 845
 nnml
 mo_constants, 125
 nobs
 mo_common_variables::period, 876
 nodata_dp
 mo_common_constants, 91
 nodata_i4
 mo_common_constants, 91
 nodata_value
 mo_common_variables::grid, 774
 nonull
 mo_string_utils, 681
 notequal_dp
 mo_utils, 709
 mo_utils::ne, 842
 mo_utils::notequal, 845
 notequal_sp
 mo_utils, 709
 mo_utils::ne, 842
 mo_utils::notequal, 845
 nout
 mo_constants, 126
 noutflxstate
 mo_mhm_constants, 248
 mo_mrm_constants, 315
 npar2_arth
 mo_corr, 138
 npar_arth
 mo_corr, 138
 nprocesses
 mo_common_variables, 120
 nroutingstates
 mo_mrm_constants, 315
 nrows
 mo_common_variables::grid, 774
 nse_dp_1d
 mo_errormeasures, 154
 mo_errormeasures::nse, 846
 nse_dp_2d
 mo_errormeasures, 155

mo_errormeasures::nse, 846
nse_dp_3d
 mo_errormeasures, 155
 mo_errormeasures::nse, 846
nse_sp_1d
 mo_errormeasures, 155
 mo_errormeasures::nse, 846
nse_sp_2d
 mo_errormeasures, 155
 mo_errormeasures::nse, 847
nse_sp_3d
 mo_errormeasures, 155
 mo_errormeasures::nse, 847
nsoilhorizons_mhm
 mo_mpr_global_variables, 285
nsoilhorizons_sm_input
 mo_global_variables, 182
nsoiltypes
 mo_mpr_global_variables, 285
nsubs
 mo_ncwrite::variable, 927
ntillhorizons
 mo_mpr_global_variables::soiltype, 890
ntimesteps_l1_et
 mo_global_variables, 182
ntimesteps_l1_neurons
 mo_global_variables, 182
ntimesteps_l1_sm
 mo_global_variables, 182
ntimestepday
 mo_common_mhm_mrm_variables, 103
number
 mo_mrm_write_fluxes_states::outputvariable, 867
 mo_write_fluxes_states::outputvariable, 862
nunique0basins
 mo_common_variables, 120
nunitnml
 mo_nml, 571
nvalues
 mo_ncwrite::attribute, 739
nvars
 mo_ncwrite, 492
objective
 mo_objective_function, 574
objective_equal_nse_lnnse
 mo_mrm_objective_function_runoff, 372
objective_et_kge_catchment_avg
 mo_objective_function, 576
objective_interface
 mo_optimization_utils::objective_interface, 850
objective_kge
 mo_mrm_objective_function_runoff, 374
objective_kge_q_et
 mo_objective_function, 577
objective_kge_q_rmse_et
 mo_objective_function, 578
objective_kge_q_rmse_tws
 mo_objective_function, 579
objective_kge_q_sm_corr
 mo_objective_function, 581
objective_lnnse
 mo_mrm_objective_function_runoff, 375
objective_multiple_gauges_kge_power6
 mo_mrm_objective_function_runoff, 377
objective_neutrons_kge_catchment_avg
 mo_objective_function, 582
objective_nse
 mo_mrm_objective_function_runoff, 378
objective_power6_nse_lnnse
 mo_mrm_objective_function_runoff, 379
objective_sm_corr
 mo_objective_function, 583
objective_sm_kge_catchment_avg
 mo_objective_function, 584
objective_sm_pd
 mo_objective_function, 586
objective_sm_sse_standard_score
 mo_objective_function, 587
objective_sse
 mo_mrm_objective_function_runoff, 380
objective_weighted_nse
 mo_mrm_objective_function_runoff, 382
of
 mo_mrm_write_fluxes_states::outputvariable, 867
 mo_netcdf::ncdataset, 822
 mo_write_fluxes_states::outputvariable, 862
oldintegration
 mo_neutrons, 564
open
 mo_netcdf::ncdataset, 822
open_netcdf
 mo_ncwrite, 478
open_nml
 mo_nml, 568
opened
 mo_netcdf::ncdataset, 822
opti_function
 mo_common_mhm_mrm_variables, 103
opti_method
 mo_common_mhm_mrm_variables, 103
optimization
 mo_optimization, 589
optimize
 mo_common_mhm_mrm_variables, 103
optimize_restart
 mo_common_mhm_mrm_variables, 103
outputflxstate
 mo_global_variables, 182
outputflxstate_mrm
 mo_mrm_global_variables, 333
p0_dp
 mo_constants, 126
p0_sp
 mo_constants, 126
p1_initstatefluxes
 mo_common_constants, 92

p2_initstatefluxes
 mo_mhm_constants, 248
 mo_mpr_constants, 264
p3_initstatefluxes
 mo_mhm_constants, 248
 mo_mpr_constants, 264
p4_initstatefluxes
 mo_mhm_constants, 248
 mo_mpr_constants, 264
p5_initstatefluxes
 mo_mhm_constants, 248
 mo_mpr_constants, 265
parameter_regularization
 mo_mrm_objective_function_runoff, 383
parent
 mo_netcdf::ncdimension, 841
pargen_anneal_dp
 mo_anneal, 78
pargen_dds_dp
 mo_anneal, 78
pargen_dp
 mo_mcmc, 227
pargennorm_dp
 mo_mcmc, 228
parstt
 mo_sce, 655
paste_char_m_m
 mo_append, 84
 mo_append::paste, 869
paste_char_m_s
 mo_append, 84
 mo_append::paste, 869
paste_char_m_v
 mo_append, 84
 mo_append::paste, 870
paste_dp_m_m
 mo_append, 84
 mo_append::paste, 870
paste_dp_m_s
 mo_append, 85
 mo_append::paste, 870
paste_dp_m_v
 mo_append, 85
 mo_append::paste, 870
paste_i4_m_m
 mo_append, 85
 mo_append::paste, 870
paste_i4_m_s
 mo_append, 85
 mo_append::paste, 870
paste_i4_m_v
 mo_append, 85
 mo_append::paste, 870
paste_i8_m_m
 mo_append, 85
 mo_append::paste, 871
paste_i8_m_s
 mo_append, 85
 mo_append::paste, 871
mo_append::paste, 871
paste_i8_m_v
 mo_append, 86
 mo_append::paste, 871
paste_lgt_m_m
 mo_append, 86
 mo_append::paste, 871
paste_lgt_m_s
 mo_append, 86
 mo_append::paste, 871
paste_lgt_m_v
 mo_append, 86
 mo_append::paste, 871
paste_sp_m_m
 mo_append, 86
 mo_append::paste, 872
paste_sp_m_s
 mo_append, 86
 mo_append::paste, 872
paste_sp_m_v
 mo_append, 86
 mo_append::paste, 872
pd_dp
 mo_spatialssimilarity, 672
 mo_spatialssimilarity::pd, 873
pd_sp
 mo_spatialssimilarity, 672
 mo_spatialssimilarity::pd, 874
peakdistribution
 mo_mrm_signatures, 411
percentile_0d_dp
 mo_percentile, 609
 mo_percentile::percentile, 874
percentile_0d_sp
 mo_percentile, 609
 mo_percentile::percentile, 874
percentile_1d_dp
 mo_percentile, 610
 mo_percentile::percentile, 874
percentile_1d_sp
 mo_percentile, 610
 mo_percentile::percentile, 875
pet_correctbyasp
 mo_mpr_pet, 287
pet_correctbylai
 mo_mpr_pet, 288
pet_hargreaves
 mo_pet, 612
pet_penman
 mo_pet, 613
pet_priestly
 mo_pet, 615
pi
 mo_constants, 126
pi_d
 mo_constants, 126
pi_dp
 mo_constants, 126

mo_template, 686
pi_sp
 mo_constants, 126
 mo_template, 687
pio2
 mo_constants, 127
pio2_d
 mo_constants, 127
pio2_dp
 mo_constants, 127
pio2_sp
 mo_constants, 127
position_nml
 mo_nml, 569
positioned
 mo_nml, 571
prepare_gridded_daily_lai_data
 mo_prepare_gridded_lai, 619
prepare_gridded_mean_monthly_lai_data
 mo_prepare_gridded_lai, 620
prepare_meteo_forcings_data
 mo_meteo_forcings, 237
priestley_taylor_alpha
 mo_mpr_pet, 290
print_startup_message
 mo_mrm_init, 341
processmatrix
 mo_common_variables, 120
project_details
 mo_common_variables, 121
psychro_dp
 mo_constants, 127
psychro_sp
 mo_constants, 127
pwp
 mo_mpr_soilmoist, 308
pwp_c
 mo_mpr_constants, 265
pwp_matpot_theta
 mo_mpr_constants, 265

q
 mo_mrm_global_variables::gaugingstation, 763
qmedian_dp
 mo_percentile, 610
 mo_percentile::qmedian, 877
qmedian_sp
 mo_percentile, 610
 mo_percentile::qmedian, 877

r_ctrper
 mo_orderpack, 603
 mo_orderpack::ctrper, 752
r_fndnth
 mo_orderpack, 603
 mo_orderpack::fndnth, 760
r_indmed
 mo_orderpack, 603
 mo_orderpack::indmed, 779

r_indnth
 mo_orderpack, 603
 mo_orderpack::indnth, 780
r_inspars
 mo_orderpack, 603
 mo_orderpack::inspar, 780
r_inssor
 mo_orderpack, 604
 mo_orderpack::inssor, 781
r_med
 mo_orderpack, 604
r_median
 mo_orderpack, 604
 mo_orderpack::omedian, 850
r_mrgref
 mo_orderpack, 604
 mo_orderpack::mrgref, 807
r_mrgrnk
 mo_orderpack, 605
 mo_orderpack::mrgrnk, 808
r_mulcnt
 mo_orderpack, 605
 mo_orderpack::mulcnt, 810
r_nearless
 mo_orderpack, 605
 mo_orderpack::nearless, 843
r_rapknr
 mo_orderpack, 605
 mo_orderpack::rapknr, 878
r_refpar
 mo_orderpack, 605
 mo_orderpack::refpar, 881
r_refsor
 mo_orderpack, 605
 mo_orderpack::refsor, 882
 mo_orderpack::sort, 894
r_rinpar
 mo_orderpack, 606
 mo_orderpack::rinpar, 883
r_rnkpar
 mo_orderpack, 606
 mo_orderpack::rnkpar, 885
r_subso
 mo_orderpack, 606
r_uniinv
 mo_orderpack, 606
 mo_orderpack::uniinv, 908
r_unipar
 mo_orderpack, 607
 mo_orderpack::unipar, 909
r_unirnk
 mo_orderpack, 607
 mo_orderpack::unirnk, 910
r_unista
 mo_orderpack, 607
 mo_orderpack::unista, 910
r_valmed
 mo_orderpack, 607

mo_orderpack::valmed, 915
 r_valnth
 mo_orderpack, 607
 mo_orderpack::valnth, 915
 RELEASES.md, 1005
 rad2deg_dp
 mo_constants, 127
 rad2deg_sp
 mo_constants, 127
 radiusearth_dp
 mo_constants, 128
 radiusearth_sp
 mo_constants, 128
 read_basin_avg_tws
 mo_read_optional_data, 631
 read_data
 mo_read_wrapper, 643
 read_dem
 mo_common_read_data, 109
 read_error
 mo_nml, 571
 read_evapotranspiration
 mo_read_optional_data, 632
 read_forcing_nc
 mo_read_forcing_nc, 623
 read_geoformation_lut
 mo_read_lut, 629
 read_grid_info
 mo_common_restart, 111
 read_header_ascii
 mo_read_spatial_data, 636
 read_lai_lut
 mo_read_lut, 630
 read_latlon
 mo_read_latlon, 627
 read_lccover
 mo_common_read_data, 110
 read_meteo_weights
 mo_global_variables, 182
 read_mrm_routing_params
 mo_mrm_read_config, 387
 read_neutrons
 mo_read_optional_data, 633
 read_restart
 mo_common_mhm_mrm_variables, 104
 read_restart_states
 mo_restart, 645
 read_soil_lut
 mo_soil_database, 664
 read_soil_moisture
 mo_read_optional_data, 634
 read_spatial_data_ascii_dp
 mo_read_spatial_data, 637
 mo_read_spatial_data::read_spatial_data_ascii,
 879
 read_spatial_data_ascii_i4
 mo_read_spatial_data, 638
 mo_read_spatial_data::read_spatial_data_ascii,
 879
 read_timeseries
 mo_read_timeseries, 639
 read_weights_nc
 mo_read_forcing_nc, 625
 readper
 mo_common_mhm_mrm_variables, 104
 realft_dp
 mo_corr, 136
 mo_corr::realft, 880
 realft_sp
 mo_corr, 136
 mo_corr::realft, 880
 reconstruct
 mo_mrm_write_fluxes_states::outputvariable, 867
 mo_write_fluxes_states::outputvariable, 862
 reg_rout
 mo_mrm_mpr, 344
 resolutionhydrology
 mo_common_variables, 121
 resolutionrouting
 mo_common_mhm_mrm_variables, 104
 rho0_dp
 mo_constants, 128
 rho0_sp
 mo_constants, 128
 right_bound
 mo_common_variables::gridremapper, 777
 rmse_dp_1d
 mo_errormeasures, 155
 mo_errormeasures::rmse, 883
 rmse_dp_2d
 mo_errormeasures, 156
 mo_errormeasures::rmse, 883
 rmse_dp_3d
 mo_errormeasures, 156
 mo_errormeasures::rmse, 884
 rmse_sp_1d
 mo_errormeasures, 156
 mo_errormeasures::rmse, 884
 rmse_sp_2d
 mo_errormeasures, 157
 mo_errormeasures::rmse, 884
 rmse_sp_3d
 mo_errormeasures, 157
 mo_errormeasures::rmse, 884
 rotate_fdir_variable
 mo_mrm_read_data, 392
 rout_space_weight
 mo_mrm_constants, 316
 routingstates
 mo_global_variables, 183
 runoff_sat_zone
 mo_runoff, 650
 runoff_unsat_zone
 mo_runoff, 650
 runoffratio

mo_mrm_signatures, 412

rzdepth
 mo_mpr_global_variables::soiltype, 890

s
 mo_netcdf::ncdimension, 841

sa_temp
 mo_common_mhm_mrm_variables, 104

sae_dp_1d
 mo_errormeasures, 158
 mo_errormeasures::sae, 885

sae_dp_2d
 mo_errormeasures, 158
 mo_errormeasures::sae, 885

sae_dp_3d
 mo_errormeasures, 159
 mo_errormeasures::sae, 886

sae_sp_1d
 mo_errormeasures, 160
 mo_errormeasures::sae, 886

sae_sp_2d
 mo_errormeasures, 160
 mo_errormeasures::sae, 886

sae_sp_3d
 mo_errormeasures, 161
 mo_errormeasures::sae, 886

sand
 mo_mpr_global_variables::soiltype, 890

sat_vap_pressure
 mo_pet, 616

satpressureslope1
 mo_mhm_constants, 248

sce
 mo_sce, 655

sce_ngs
 mo_common_mhm_mrm_variables, 104

sce_npg
 mo_common_mhm_mrm_variables, 104

sce_nps
 mo_common_mhm_mrm_variables, 104

secday_dp
 mo_constants, 128

secday_sp
 mo_constants, 128

seed
 mo_common_mhm_mrm_variables, 105

selectcalendar
 mo Julian, 218

separator
 mo_string_utils, 684

set_basin_indices
 mo_grid, 189

set_land_cover_scenes_id
 mo_common_read_config, 107

set_netcdf
 mo_set_netcdf_outputs, 660

set_optional
 mo_sce.f90, 990

setattribute

 mo_netcdf::ncdataset, 818
 mo_netcdf::ncdimension, 834

setcalendarinteger
 mo Julian, 219
 mo Julian::setcalendar, 887

setcalendarstring
 mo Julian, 220
 mo Julian::setcalendar, 887

setdata
 mo_netcdf::ncdimension, 834

setdata1df32
 mo_netcdf, 531
 mo_netcdf::ncdimension, 835

setdata1df64
 mo_netcdf, 531
 mo_netcdf::ncdimension, 835

setdata1di16
 mo_netcdf, 532
 mo_netcdf::ncdimension, 835

setdata1di32
 mo_netcdf, 532
 mo_netcdf::ncdimension, 835

setdata1di64
 mo_netcdf, 533
 mo_netcdf::ncdimension, 835

setdata1di8
 mo_netcdf, 533
 mo_netcdf::ncdimension, 835

setdata2df32
 mo_netcdf, 534
 mo_netcdf::ncdimension, 836

setdata2df64
 mo_netcdf, 534
 mo_netcdf::ncdimension, 836

setdata2di16
 mo_netcdf, 535
 mo_netcdf::ncdimension, 836

setdata2di32
 mo_netcdf, 535
 mo_netcdf::ncdimension, 836

setdata2di64
 mo_netcdf, 536
 mo_netcdf::ncdimension, 836

setdata2di8
 mo_netcdf, 536
 mo_netcdf::ncdimension, 836

setdata3df32
 mo_netcdf, 537
 mo_netcdf::ncdimension, 836

setdata3df64
 mo_netcdf, 537
 mo_netcdf::ncdimension, 836

setdata3di16
 mo_netcdf, 538
 mo_netcdf::ncdimension, 836

setdata3di32
 mo_netcdf, 538
 mo_netcdf::ncdimension, 837

setdata3di64
 mo_ncdf, 539
 mo_ncdf::ncdimension, 837
setdata3di8
 mo_ncdf, 539
 mo_ncdf::ncdimension, 837
setdata4df32
 mo_ncdf, 540
 mo_ncdf::ncdimension, 837
setdata4df64
 mo_ncdf, 540
 mo_ncdf::ncdimension, 837
setdata4di16
 mo_ncdf, 541
 mo_ncdf::ncdimension, 837
setdata4di32
 mo_ncdf, 541
 mo_ncdf::ncdimension, 837
setdata4di64
 mo_ncdf, 542
 mo_ncdf::ncdimension, 837
setdata4di8
 mo_ncdf, 542
 mo_ncdf::ncdimension, 837
setdata5df32
 mo_ncdf, 543
 mo_ncdf::ncdimension, 838
setdata5df64
 mo_ncdf, 543
 mo_ncdf::ncdimension, 838
setdata5di16
 mo_ncdf, 544
 mo_ncdf::ncdimension, 838
setdata5di32
 mo_ncdf, 544
 mo_ncdf::ncdimension, 838
setdata5di64
 mo_ncdf, 545
 mo_ncdf::ncdimension, 838
setdata5di8
 mo_ncdf, 545
 mo_ncdf::ncdimension, 838
setdatascalarf32
 mo_ncdf, 546
 mo_ncdf::ncdimension, 838
setdatascalarf64
 mo_ncdf, 546
 mo_ncdf::ncdimension, 838
setdatascalari16
 mo_ncdf, 547
 mo_ncdf::ncdimension, 838
setdatascalari32
 mo_ncdf, 547
 mo_ncdf::ncdimension, 839
setdatascalari64
 mo_ncdf, 547
 mo_ncdf::ncdimension, 839
setdatascalari8
 mo_ncdf, 548
 mo_ncdf::ncdimension, 839
setdimension
 mo_ncdf, 548
 mo_ncdf::ncdataset, 819
setfillvalue
 mo_ncdf::ncdimension, 839
setglobalattributechar
 mo_ncdf, 549
 mo_ncdf::ncdataset, 819
setglobalattributef32
 mo_ncdf, 549
 mo_ncdf::ncdataset, 819
setglobalattributef64
 mo_ncdf, 549
 mo_ncdf::ncdataset, 819
setglobalattributei16
 mo_ncdf, 550
 mo_ncdf::ncdataset, 819
setglobalattributei32
 mo_ncdf, 550
 mo_ncdf::ncdataset, 819
setglobalattributei64
 mo_ncdf, 551
 mo_ncdf::ncdataset, 820
setglobalattributei8
 mo_ncdf, 551
 mo_ncdf::ncdataset, 820
setup_description
 mo_common_variables, 121
setvariable
 mo_ncdf::ncdataset, 820
setvariableattributechar
 mo_ncdf, 551
 mo_ncdf::ncdimension, 839
setvariableattributef32
 mo_ncdf, 552
 mo_ncdf::ncdimension, 839
setvariableattributef64
 mo_ncdf, 552
 mo_ncdf::ncdimension, 840
setvariableattributei16
 mo_ncdf, 553
 mo_ncdf::ncdimension, 840
setvariableattributei32
 mo_ncdf, 553
 mo_ncdf::ncdimension, 840
setvariableattributei64
 mo_ncdf, 553
 mo_ncdf::ncdimension, 840
setvariableattributei8
 mo_ncdf, 554
 mo_ncdf::ncdimension, 840
setvariablefillvaluef32
 mo_ncdf, 554
 mo_ncdf::ncdimension, 840
setvariablefillvaluef64
 mo_ncdf, 554

mo_netcdf::ncdimension, 840
setvariablefillvaluei16
 mo_netcdf, 554
 mo_netcdf::ncdimension, 840
setvariablefillvaluei32
 mo_netcdf, 555
 mo_netcdf::ncdimension, 840
setvariablefillvaluei64
 mo_netcdf, 555
 mo_netcdf::ncdimension, 841
setvariablefillvaluei8
 mo_netcdf, 555
 mo_netcdf::ncdimension, 841
setvariablewithids
 mo_netcdf, 555
 mo_netcdf::ncdataset, 821
setvariablewithnames
 mo_netcdf, 556
 mo_netcdf::ncdataset, 821
setvariablewithtypes
 mo_netcdf, 557
 mo_netcdf::ncdataset, 821
sigma_dp
 mo_constants, 128
sigma_sp
 mo_constants, 128
simper
 mo_common_mhm_mrm_variables, 105
simulation_type
 mo_common_variables, 121
single_objective_runoff
 mo_mrm_objective_function_runoff, 384
skewness_dp
 mo_moment, 260
 mo_moment::skewness, 888
skewness_sp
 mo_moment, 261
 mo_moment::skewness, 888
slope_satpressure
 mo_pet, 617
snow_acc_melt_param
 mo_multi_param_reg, 444
snow_accum_melt
 mo_snow_accum_melt, 661
soil_moisture
 mo_soil_moisture, 667
soildb
 mo_mpr_global_variables, 285
solarconst_dp
 mo_constants, 129
solarconst_sp
 mo_constants, 129
sort_index_dp
 mo_orderpack, 607
 mo_orderpack::sort_index, 894
sort_index_i4
 mo_orderpack, 607
 mo_orderpack::sort_index, 895
sort_index_sp
 mo_orderpack, 608
 mo_orderpack::sort_index, 895
sort_matrix
 mo_sce, 659
sp
 mo_kind, 223
sp2str
 mo_string_utils, 682
 mo_string_utils::num2str, 848
spatial_aggregation_3d
 mo_spatial_agg_disagg_forcing, 670
 mo_spatial_agg_disagg_forcing::spatial_aggregation, 895
spatial_aggregation_4d
 mo_spatial_agg_disagg_forcing, 670
 mo_spatial_agg_disagg_forcing::spatial_aggregation, 896
spatial_disaggregation_3d
 mo_spatial_agg_disagg_forcing, 670
 mo_spatial_agg_disagg_forcing::spatial_disaggregation, 897
spatial_disaggregation_4d
 mo_spatial_agg_disagg_forcing, 671
 mo_spatial_agg_disagg_forcing::spatial_disaggregation, 897
spc
 mo_kind, 223
specheatet_dp
 mo_constants, 129
specheatet_sp
 mo_constants, 129
special_value_dp
 mo_utils, 709
 mo_utils::special_value, 898
special_value_sp
 mo_utils, 710
 mo_utils::special_value, 898
splitstring
 mo_string_utils, 682
sqrt2
 mo_constants, 129
sqrt2_d
 mo_constants, 129
sqrt2_dp
 mo_constants, 129
sqrt2_sp
 mo_constants, 130
sse_dp_1d
 mo_errormeasures, 162
 mo_errormeasures::sse, 902
sse_dp_2d
 mo_errormeasures, 162
 mo_errormeasures::sse, 902
sse_dp_3d
 mo_errormeasures, 163
 mo_errormeasures::sse, 902
sse_sp_1d

mo_errormeasures, 164
 mo_errormeasures::sse, 902
 sse_sp_2d
 mo_errormeasures, 164
 mo_errormeasures::sse, 902
 sse_sp_3d
 mo_errormeasures, 165
 mo_errormeasures::sse, 902
 standard_score_dp
 mo_standard_score, 673
 mo_standard_score::standard_score, 904
 standard_score_sp
 mo_standard_score, 674
 mo_standard_score::standard_score, 904
 start
 mo_ncwrite::variable, 927
 startswith
 mo_string_utils, 682
 status
 mo_timer, 699
 stboltzmann
 mo_mhm_constants, 248
 stddev_dp
 mo_moment, 261
 mo_moment::stddev, 904
 stddev_sp
 mo_moment, 261
 mo_moment::stddev, 904
 steps
 mo_mrm_write_fluxes_states::outputdataset, 853
 mo_write_fluxes_states::outputdataset, 857
 store
 mo_mrm_write_fluxes_states::outputdataset, 853
 mo_mrm_write_fluxes_states::outputvariable, 867
 mo_write_fluxes_states::outputdataset, 857
 mo_write_fluxes_states::outputvariable, 862
 str2num
 mo_string_utils, 683
 swap_1d_dpc
 mo_corr, 137
 mo_corr::swap, 905
 swap_1d_spc
 mo_corr, 137
 mo_corr::swap, 905
 swap_vec_dp
 mo_utils, 710
 mo_utils::swap, 906
 swap_vec_i4
 mo_utils, 710
 mo_utils::swap, 906
 swap_vec_sp
 mo_utils, 711
 mo_utils::swap, 906
 swap_xy_dp
 mo_utils, 711
 mo_utils::swap, 906
 swap_xy_i4
 mo_utils, 711

mo_utils::swap, 906
 swap_xy_sp
 mo_utils, 711
 mo_utils::swap, 907
 t0_dp
 mo_constants, 130
 t0_sp
 mo_constants, 130
 tabularintegralafast
 mo_neutrons, 565
 temporal_disagg_forcing
 mo_temporal_disagg_forcing, 689
 tetens_c1
 mo_mhm_constants, 248
 tetens_c2
 mo_mhm_constants, 249
 tetens_c3
 mo_mhm_constants, 249
 the
 mo_mrm_write_fluxes_states::outputvariable, 867
 mo_netcdf::ncdataset, 822
 mo_netcdf::ncdimension, 841, 842
 mo_write_fluxes_states::outputvariable, 862
 thetafc
 mo_mpr_global_variables::soiltype, 890
 thetafc_till
 mo_mpr_global_variables::soiltype, 890
 thetapw
 mo_mpr_global_variables::soiltype, 890
 thetapw_till
 mo_mpr_global_variables::soiltype, 891
 thetas
 mo_mpr_global_variables::soiltype, 891
 thetas_till
 mo_mpr_global_variables::soiltype, 891
 tillagedepth
 mo_mpr_global_variables, 285
 time
 mo_mrm_write_fluxes_states::outputdataset, 853
 mo_write_fluxes_states::outputdataset, 857
 timer_check
 mo_timer, 691
 timer_clear
 mo_timer, 692
 timer_get
 mo_timer, 693
 timer_print
 mo_timer, 694
 timer_start
 mo_timer, 695
 timer_stop
 mo_timer, 696
 timers_init
 mo_timer, 697
 timestep
 mo_common_mhm_mrm_variables, 105
 timestep_et_input
 mo_global_variables, 183

timestep_lai_input
 mo_mpr_global_variables, 285

timestep_model_inputs
 mo_global_variables, 183

timestep_model_outputs
 mo_global_variables, 183

timestep_model_outputs_mrm
 mo_mrm_global_variables, 333

timestep_neutrons_input
 mo_global_variables, 183

timestep_sm_input
 mo_global_variables, 183

to
 mo_mrm_write_fluxes_states::outputdataset, 854
 mo_mrm_write_fluxes_states::outputvariable, 867
 mo_write_fluxes_states::outputdataset, 858
 mo_write_fluxes_states::outputvariable, 862

tolower
 mo_string_utils, 683

toupper
 mo_string_utils, 684

twopi
 mo_constants, 130

twopi_d
 mo_constants, 130

twopi_dp
 mo_constants, 130

twopi_sp
 mo_constants, 130

twothird_dp
 mo_constants, 131

twothird_sp
 mo_constants, 131

tws
 mo_global_variables::twsstructure, 908

uaspect
 mo_mpr_file, 274

uconfig
 mo_common_file, 93
 mo_mrm_file, 321

ud
 mo_mpr_global_variables::soiltype, 891

udaily_discharge
 mo_mrm_file, 321

udefoutput
 mo_file, 168
 mo_mrm_file, 322

udem
 mo_common_file, 93

udischarge
 mo_mrm_file, 322

ufacc
 mo_mrm_file, 322

ufdir
 mo_mrm_file, 322

ugaugeloc
 mo_mrm_file, 322

ugeolut

mo_mpr_file, 274

uhydrogeoclass
 mo_mpr_file, 274

ulaiclass
 mo_mpr_file, 275

ulailut
 mo_mpr_file, 275

ulcoverclass
 mo_common_file, 94

umeteo
 mo_mpr_file, 275

umeteo_header
 mo_mpr_file, 275

unamelist_mhm
 mo_file, 168

unamelist_mhm_param
 mo_file, 168

unamelist_mpr
 mo_mpr_file, 275

unamelist_mpr_param
 mo_mpr_file, 275

unamelist_mrm
 mo_mrm_file, 322

unamelist_param_mrm
 mo_mrm_file, 322

unlimited
 mo_ncwrite::variable, 927

unpack_field_and_write_1d_dp
 mo_mpr_restart, 294
 mo_mpr_restart::unpack_field_and_write, 911
 mo_restart, 646
 mo_restart::unpack_field_and_write, 913

unpack_field_and_write_1d_i4
 mo_mpr_restart, 294
 mo_mpr_restart::unpack_field_and_write, 912
 mo_restart, 646
 mo_restart::unpack_field_and_write, 913

unpack_field_and_write_2d_dp
 mo_mpr_restart, 294
 mo_mpr_restart::unpack_field_and_write, 912
 mo_restart, 647
 mo_restart::unpack_field_and_write, 914

unpack_field_and_write_3d_dp
 mo_mpr_restart, 295
 mo_mpr_restart::unpack_field_and_write, 912
 mo_restart, 647
 mo_restart::unpack_field_and_write, 914

uopti
 mo_common_mhm_mrm_file, 96

uopti_nml
 mo_common_mhm_mrm_file, 96

updatedataset
 mo_mrm_write_fluxes_states, 429
 mo_mrm_write_fluxes_states::outputdataset, 852
 mo_write_fluxes_states, 720
 mo_write_fluxes_states::outputdataset, 856

updatevariable
 mo_mrm_write_fluxes_states, 430

mo_mrm_write_fluxes_states::outputvariable, 865, 868
 mo_write_fluxes_states, 721
 mo_write_fluxes_states::outputvariable, 860, 863
 upper_bound
 mo_common_variables::gridremapper, 777
 upscale_arithmetic_mean
 mo_upscaling_operators, 701
 upscale_geometric_mean
 mo_upscaling_operators, 703
 upscale_harmonic_mean
 mo_upscaling_operators, 704
 upscale_p_norm
 mo_upscaling_operators, 705
 uslope
 mo_mpr_file, 276
 usoil_database
 mo_mpr_file, 276
 usoilclass
 mo_mpr_file, 276
 utws
 mo_file, 168

v

val

values

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

mo_ncwrite, 492

mo_kind::sprs2_dp, 900

mo_kind::sprs2_sp, 901

mo_ncwrite::attribute, 739

mo_ncwrite, 480

mo_ncwrite::var2nc, 917

mo_ncwrite, 481

mo_ncwrite::var2nc, 918

mo_ncwrite, 482

mo_ncwrite::var2nc, 918

mo_ncwrite, 482

mo_ncwrite::var2nc, 918

mo_ncwrite, 483

mo_ncwrite::var2nc, 918

mo_ncwrite, 484

mo_ncwrite::var2nc, 919

mo_ncwrite, 484

mo_ncwrite::var2nc, 919

mo_ncwrite, 485

mo_ncwrite::var2nc, 919

mo_ncwrite, 486

mo_ncwrite::var2nc, 920

mo_ncwrite, 486

mo_ncwrite::var2nc, 920

mo_ncwrite::var2nc, 920

mo_ncwrite::var2nc, 487

mo_ncwrite::var2nc, 920

mo_ncwrite::var2nc, 921

mo_ncwrite::var2nc, 488

mo_ncwrite::var2nc, 921

mo_ncwrite::var2nc, 488

mo_ncwrite::var2nc, 921

mo_ncwrite, 489

mo_ncwrite::var2nc, 921

mo_ncwrite, 490

mo_ncwrite::var2nc, 922

variable

variables

variables_alloc

variables_alloc_routing

variables_default_init

variables_default_init_routing

variance_dp

variance_sp

varid

varnametotalrunoff

vars

version

version_date

vgenuchten_sandtresh

vgenuchten_c1

vgenuchten_c10

vgenuchten_c11

mo_init_states, 190

mo_mrm_init, 341

mo_init_states, 191

mo_mrm_init, 342

mo_moment, 261

mo_moment::variance, 928

mo_moment, 261

mo_moment::variance, 928

mo_ncwrite::variable, 928

mo_mrm_global_variables, 333

mo_mrm_write_fluxes_states::outputdataset, 854

mo_write_fluxes_states::outputdataset, 858

mo_file, 169

mo_mpr_file, 276

mo_mrm_file, 323

mo_file, 169

mo_mpr_file, 276

mo_mrm_file, 323

mo_mpr_constants, 265

mo_mpr_constants, 265

mo_mpr_constants, 265

mo_file, 169

mo_mpr_constants, 265
vgenuchtenn_c12
 mo_mpr_constants, 266
vgenuchtenn_c13
 mo_mpr_constants, 266
vgenuchtenn_c14
 mo_mpr_constants, 266
vgenuchtenn_c15
 mo_mpr_constants, 266
vgenuchtenn_c16
 mo_mpr_constants, 266
vgenuchtenn_c17
 mo_mpr_constants, 266
vgenuchtenn_c18
 mo_mpr_constants, 266
vgenuchtenn_c2
 mo_mpr_constants, 266
vgenuchtenn_c3
 mo_mpr_constants, 267
vgenuchtenn_c4
 mo_mpr_constants, 267
vgenuchtenn_c5
 mo_mpr_constants, 267
vgenuchtenn_c6
 mo_mpr_constants, 267
vgenuchtenn_c7
 mo_mpr_constants, 267
vgenuchtenn_c8
 mo_mpr_constants, 267
vgenuchtenn_c9
 mo_mpr_constants, 267

warmingdays
 mo_common_mhm_mrm_variables, 105
warpper
 mo_common_mhm_mrm_variables, 105
wd
 mo_mpr_global_variables::soiltype, 891
wflag
 mo_ncwrite::variable, 928
which
 mo_mrm_write_fluxes_states::outputvariable, 868
 mo_write_fluxes_states::outputvariable, 863
windmeasheight
 mo_mpr_constants, 268
wnse_dp_1d
 mo_errormeasures, 166
 mo_errormeasures::wnse, 929
wnse_dp_2d
 mo_errormeasures, 166
 mo_errormeasures::wnse, 929
wnse_dp_3d
 mo_errormeasures, 166
 mo_errormeasures::wnse, 929
wnse_sp_1d
 mo_errormeasures, 166
 mo_errormeasures::wnse, 929
wnse_sp_2d
 mo_errormeasures, 166

mo_errormeasures::wnse, 929
wnse_sp_3d
 mo_errormeasures, 166
 mo_errormeasures::wnse, 930
write
 mo_mrm_write_fluxes_states::outputdataset, 854
 mo_write_fluxes_states::outputdataset, 858
write_best_final
 mo_sce.f90, 990
write_best_intermediate
 mo_sce.f90, 991
write_configfile
 mo_mrm_write, 421
 mo_write_ascii, 712
write_daily_obs_sim_discharge
 mo_mrm_write, 422
write_dynamic_ncdf
 mo_ncwrite, 490
write_eff_params
 mo_mpr_restart, 295
write_grid_info
 mo_common_restart, 113
write_mpr_restart_files
 mo_mpr_restart, 296
write_optifile
 mo_write_ascii, 713
write_optinamelist
 mo_write_ascii, 714
write_population
 mo_sce.f90, 991
write_restart
 mo_common_variables, 121
write_restart_files
 mo_restart, 647
write_static_ncdf
 mo_ncwrite, 491
write_termination_case
 mo_sce.f90, 991
writes
 mo_mrm_write_fluxes_states::outputvariable, 868
 mo_write_fluxes_states::outputvariable, 863
writetimestep
 mo_mrm_write_fluxes_states, 431
 mo_mrm_write_fluxes_states::outputdataset, 852
 mo_write_fluxes_states, 722
 mo_write_fluxes_states::outputdataset, 856
writevariableattributes
 mo_mrm_write_fluxes_states, 431
 mo_write_fluxes_states, 723
writevariabletimestep
 mo_mrm_write_fluxes_states, 432
 mo_mrm_write_fluxes_states::outputvariable, 865
 mo_write_fluxes_states, 723
 mo_write_fluxes_states::outputvariable, 860
writing
 mo_mrm_write_fluxes_states::outputvariable, 868
 mo_write_fluxes_states::outputvariable, 863
written

mo_mrm_write_fluxes_states::outputdataset, 854
 mo_write_fluxes_states::outputdataset, 858

 x
 mo_common_variables::grid, 775
 xllcorner
 mo_common_variables::grid, 775
 xor4096d_0d
 mo_xor4096, 725
 mo_xor4096::xor4096, 930
 xor4096d_1d
 mo_xor4096, 725
 mo_xor4096::xor4096, 930
 xor4096f_0d
 mo_xor4096, 726
 mo_xor4096::xor4096, 930
 xor4096f_1d
 mo_xor4096, 726
 mo_xor4096::xor4096, 931
 xor4096gd_0d
 mo_xor4096, 726
 mo_xor4096::xor4096g, 932
 xor4096gd_1d
 mo_xor4096, 726
 mo_xor4096::xor4096g, 932
 xor4096gf_0d
 mo_xor4096, 726
 mo_xor4096::xor4096g, 932
 xor4096gf_1d
 mo_xor4096, 727
 mo_xor4096::xor4096g, 932
 xor4096l_0d
 mo_xor4096, 727
 mo_xor4096::xor4096, 931
 xor4096l_1d
 mo_xor4096, 727
 mo_xor4096::xor4096, 931
 xor4096s_0d
 mo_xor4096, 727
 mo_xor4096::xor4096, 931
 xor4096s_1d
 mo_xor4096, 727
 mo_xor4096::xor4096, 931
 xtype
 mo_ncwrite::attribute, 739
 mo_ncwrite::variable, 928

 y
 mo_common_variables::grid, 775
 year_counter
 mo_mrm_write, 424
 yeardays
 mo_common_constants, 92
 mo_constants, 131
 yearmonths
 mo_common_constants, 92
 mo_constants, 131
 yearmonths_i4
 mo_common_constants, 92